МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
« ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

ДО ВИКОНАННЯ ТА ОФОРМЛЕННЯ КУРСОВОГО ПРОЕКТУ
З КУРСУ
Системне програмування

для студентів спеціальностей:
7.05010201 «Комп'ютерні системи та мережі»,
7.05010202 «Системне програмування»,
7.05010203 «Спеціалізовані комп'ютерні системи»

Затверджено
редакційно-видавничою
радою університету,
протокол № 1 від 20.06.2013 р.

Харків
НТУ "ХПІ"
2013

Методичні вказівки до виконання та оформлення курсового проекту з курсу «Системне програмування» для студентів спеціальностей: 7.05010201 «Комп'ютерні системи та мережі», 7.05010202 «Системне програмування», 7.05010203 «Спеціалізовані комп'ютерні системи» / уклад. О. М. Рисований – Х. : НТУ «ХПІ», 2013. – 184 с.

Укладач О. М. Рисований

Рецензент А. І. Поворознюк

Кафедра обчислювальної техніки та програмування

ЗМІСТ

ВСТУП

При написанні програми програміст повинен зробити її конкурентоспроможною. Для цього необхідно запрограмувати не тільки ефективний алгоритм виконання або обробки інформації, але й надати інтерфейс користувача, за допомогою якого виконується керування програмою [1, 4-6]. Замовник спочатку сприймає інтерфейс програми і від його професійного сучасного вигляду формується перше враження від розробки. Тому віконному програмуванню відводиться велике значення.

Курсовий проект з навчальної дисципліни «Системне програмування» виконується студентами кафедри «Обчислювальна техніка та програмування» у 2-му семестрі.

Мета курсового проектування – закріпити теоретичні знання з проектування програм на мові асемблер, утиліт, програмування драйверів, дослідження показників продуктивності ОС, аналіз процесів, які виникають у її ядрі та між різними менеджерами і диспетчерами, навчити студентів самостійно застосовувати отримані знання для постановки та розв’язання задач з проектування системних програм різного призначення.

Завдання проектування – самостійне вирішення конкретної професійної проблеми у зазначеному напрямку.

Завершена курсова робота (проект) подається виконавцем керівнику не пізніше як за п’ять днів до захисту і містить текстуальну частину та графічний матеріал, що її ілюструє.

Курсовий проект з висновками керівника повертається виконавцю не пізніше як за день до захисту. У тих випадках, коли курсовий проект не відповідає завданню або містить принципові помилки, він повертається виконавцю для доопрацювання.

1. ІНТЕРФЕЙС GDI+

GDI+ (Graphic Device Interface+ – інтерфейс графічних пристроїв) – це підсистема Microsoft Windows XP та .NET Server, що забезпечує виведення графічної інформації на екрани та принтери. GDI+ є наступником GDI, інтерфейсу графічних пристроїв, що включена в ранні версії Windows.

Можливості GDI+-технології (http://www.rsdn.ru/article/gdi/ gdiplus1. xml) такі:

– градієнтне зафарбовування;

– підтримка прозорості;

– використання сплайнів: окрім кривих Безьє, підтримується новий вигляд кривих – так звані сплайни, які імітують поведінку натягнутої і зігнутої сталевої смуги. Сплайни є гладкими кривими;

– використання шляхів: шляхи існують незалежно від контексту рисування і є засобом створення складних векторних об'єктів;

– використання координатних перетворень: об'єкт Matrix дозволяє здійснювати операції повороту, перенесення, масштабування і віддзеркалення об'єктів GDI+;

– використання регіонів: на відміну від GDI, регіони не прив'язані до координат пристрою і підпорядковуються координатним перетворенням;

– використання растрів: підтримуються растрі з накладенням зовнішнього альфа-каналу (для прозорості), масштабуванням, розтягуванням, спотворенням і поворотом растрів. При цьому можна встановити режими відображення окремих пікселів;

– підтримка популярних форматів графічних файлів.

1.1. Інтерфейс керованих класів

Доступ до API-інтерфейсу в GDI+ здійснюється через набір класів, з яких створюється керований код. Цей набір класів називається інтерфейсом керованих класів GDI+ та складається з таких просторів імен:

System.Drawing – доступ до основних графічних функцій GDI+;

System.Drawing.Drawing2D – розширений набір засобів для створення двовимірної і векторної графіки;

System.Drawing.Imaging – розширений набір графічних функцій GDI+;

System.Drawing.Text – розширений набір функцій друкарень GDI+;

System.Drawing.Printing – служби, пов'язані з друком;

TextRenderer – функції для рисування тексту GDI і управління його параметрами.

1.2. Методи рисування за допомогою графічних об'єктів

Клас Graphics інтерфейсу GDI+ містить такі методи для рисування елементів: DrawLine (прямі лінії), DrawRectangle (прямокутники), DrawEllipse (еліпси), DrawPolygon (багатокутники), DrawArc (дуги), DrawCurve (фундаментальні сплайни) та DrawBezier (сплайни Безьє). Кожен з цих методів ***переобтяжений***, а це означає, що кожен метод може отримувати різні набори параметрів. Наприклад, один варіант методу DrawLine отримує об'єкт Pen і чотири цілі числа, а інший варіант методу DrawLine (з такою ж назвою) отримує об'єкт Pen та два об'єкти Point.

Крім перерахованих раніше методів для рисування ліній, прямокутників і сплайнів Безье існують допоміжні методи, що виконують рисування декількох подібних елементів (ліній, прямокутників або сплайнів) за один виклик: DrawLines, DrawRectangles та DrawBeziers. Для методу DrawCurve також існує допоміжний метод DrawClosedCurve, що замикає криву шляхом з'єднання останньої крапки кривої з першою.

Всі призначені для рисування методи класу Graphics використовують об'єкт Pen. Щоб нарисувати який-небудь елемент, потрібно створити як мінімум два об'єкти: об'єкт Graphics і об'єкт Pen. Об'єкт Pen призначений для зберігання таких атрибутів рисованого елемента, як ширина лінії і колір. Об'єкт Pen передається в кожен метод рисування як один із аргументів:

```
invoke GdipCreateFromHDC,hdc,addr graphics
invoke GdipCreatePen1,07F0000FFh,\;7F - прозорість, RGB
      rWidth,UnitPixel,addr pen
```

1.3. Ініціалізація та завершення

Перш ніж почати використовувати класи і функції Gdi+, необхідно ініціалізувати цю бібліотеку. Для цього на початку програми потрібно помістити виклик функції GdiplusStartup:

```
Status GdiplusStartup( ULONG_PTR* token,
   const GdiplusStartupInput* input,
```

```
    GdiplusStartupOutput* output )
```

Поля структури GdiplusStartupInput управляють різними аспектами ініціалізації: зокрема, можна задати функцію, яка викликатиметься при виникненні помилок, або перехоплюватиме всі звернення до функцій Gdi+. Конструктор за умовчанням виконує ініціалізацію структури GdiplusStartupInput з параметрами, достатніми для більшості випадків. При цьому як вихідний параметр output можна задати NULL. Для masm фрагмент коду може бути таким:

```
GdiplusStartupInput STRUCT
   GdiplusVersion DWORD ?
   DebugEventCallback DWORD ?
   SuppressBackgroundThread DWORD ?
   SuppressExternalCodecs DWORD ?
GdiplusStartupInput ENDS
…
     LOCAL gdiplusStartupInput:GdiplusStartupInput
     LOCAL gdiplusToken:DWORD
mov gdiplusStartupInput.GdiplusVersion,1 ; ініціалізація бібліотеки
and gdiplusStartupInput.DebugEventCallback,0
and gdiplusStartupInput.SuppressBackgroundThread,0
and gdiplusStartupInput.SuppressExternalCodecs,0
invoke GdiplusStartup,addr gdiplusToken,addr gdiplusStartupInput,0
```

Вихідний параметр token необхідно зберегти.

Для завершення роботи з бібліотекою необхідно викликати функцію GdiplusShutdown:

```
VOID GdiplusShutdown(
     ULONG_PTR token );
```

Тут як параметр необхідно передати те саме число, яке повернула функція GdiplusStartup в параметрі знак.

1.4. Клас Graphics

Конкретний об'єкт – представник класу Graphics надається у вигляді посилання методами-обробниками подій або створюється в ході виконання ряду методів відносно конкретних об'єктів, що володіють "поверхнями рисування" (клієнтська область форми, кнопки, панелі, бітова матриця).

Список членів класу Graphics наведено в табл. 1.1.

Таблиця 1.1 – Список класу Graphics

Список членів класу Graphics	Призначення
Clip	Отримує або задає об'єкт Region, що обмежує область рисування даного об'єкта Graphics
ClipBounds	Отримує структуру RectangleF, яка містить у собі вирізану область даного об'єкта Graphics
CompositingMode	Набуває значення, яке задає порядок рисування складних зображень у даному об'єкті Graphics
CompositingQuality	Отримує або задає якість відображення складних зображень, які виводяться у даному об'єкті Graphics
DpiX	Отримує горизонтальний дозвіл даного об'єкта Graphics
DpiY	Отримує вертикальний дозвіл даного об'єкта Graphics
InterpolationMode	Отримує або задає режим вставки, пов'язаний з даним об'єктом Graphics
IsClipEmpty	Набуває значення, яке вказує, чи є вирізана область даного об'єкта Graphics порожньою
IsVisibleClipEmpty	Набуває значення, яке вказує, чи є видима вирізана область даного об'єкта Graphics порожньою
PageScale	Отримує або задає масштабування між універсальними одиницями і одиницями сторінки для даного об'єкта Graphics
PageUnit	Отримує або задає одиницю вимірювання для координат сторінки даного об'єкта Graphics
PixelOffsetMode	Отримує або задає значення, яке задає порядок зсуву крапок під час відображення даного об'єкта Graphics
RenderingOrigin	Отримує або задає початкове заповнення даного об'єкта Graphics для згладжування колірних переходів і для штрихування
SmoothingMode	Отримує або задає якість заповнення для даного об'єкта Graphics
TextContrast	Отримує або задає значення корекції яскравості для відображення тексту
TextRenderingHint	Отримує або задає режим заповнення для тексту, пов'язаного з даним об'єктом Graphics

Продовження табл. 1.1

Transform	Отримує або задає універсальне перетворення для даного об'єкта Graphics
VisibleClipBounds	Отримує або задає робочий прямокутник видимої вирізаної області об'єкта Graphics

Відкриті методи класу Graphics наведено в табл. 1.2.

Таблиця 1.2 – Відкриті методи класу Graphics

Назва методів класу Graphics	Призначення
AddMetafile-Comment	Додає коментар до потокового об'єкта Metafile
Begin-Container	Переобтяжений. Зберігає графічний контейнер, що містить потоковий стан даного об'єкта Graphics, а потім відкриває і використовує новий графічний контейнер
Clear	Очищає всю поверхню для рисування і виконує заливку поверхні вказаним кольором фону
CreateObjRef	Створює об'єкт, який містить всю необхідну інформацію для створення прокси-серверу при комунікації з видаленими об'єктами
Dispose	Звільняє всі ресурси, використовувані даним об'єктом Graphics
DrawArc	Переобтяжений. Рисує дугу, яка є частиною еліпса, заданого парою координат, шириною і висотою
DrawBezier	Переобтяжений. Будує криву Безьє, визначувану чотирма структурами Point
DrawBeziers	Переобтяжений. Формує набір кривих Безьє з масиву структур Point
DrawClosed-Curve	Переобтяжений. Будує замкнуту фундаментальну криву, визначувану масивом структур Point
DrawCurve	Переобтяжений. Будує замкнуту фундаментальну криву через точки вказаного масиву структур Point
DrawEllipse	Переобтяжений. Формує еліпс, який визначається обмежуючим прямокутником, заданим за допомогою пари координат – ширини і висоти
DrawIcon	Переобтяжений. Формує зображення, яке подане указаним об'єктом Icon, розташованим за вказаними координатами
DrawIcon-Unstretched	Формує зображення, подане вказаним об'єктом Icon, не масштабуючи його

Продовження табл. 1.2

DrawImage	Переобтяжений. Рисує заданий об'єкт Image у заданому місці, використовуючи початковий розмір
DrawImage-Unscaled	Переобтяжений. Рисує задане зображення, використовуючи його початковий фактичний розмір, у розташуванні, заданому парою координат
DrawLine	Переобтяжений. Проводить лінію, що сполучає дві крапки, визначені парами координат
DrawLines	Переобтяжений. Формує набір сегментів лінії, які сполучають масив структур Point
DrawPath	Рисує об'єкт Graphicspath
DrawPie	Переобтяжений. Рисує сектор, визначений еліпсом, який заданий парою координат, шириною, висотою і двома радіальними лініями
DrawPolygon	Переобтяжений. Рисує багатокутник, визначуваний масивом структур Point
Draw-Rectangles	Переобтяжений. Рисує набір прямокутників, визначуваних структурою Rectangle
DrawString	Переобтяжений. Створює текстовий рядок у заданому місці з указаними об'єктами Brush і Font
EndContainer	Закриває потоковий графічний контейнер і відновлює стан даного об'єкта Graphics, який було збережено при виклику методу BeginContainer
EnumerateMetafile	Переобтяжений. Відправляє записи указаного об'єкта Metafile окремо методу зворотного виклику, який відображає їх у заданій точці
Enumerate-Metafile	Переобтяжений. Відправляє записи указаного об'єкта Metafile окремо методу зворотного виклику, який відображає їх у заданій точці
Equals	Переобтяжений. Визначає, чи рівні два екземпляри Object
ExcludeClip	Переобтяжений. Оновлює вирізану область даного об'єкта Graphics, щоб виключити з неї частину, визначену структурою Rectangle
FillClosed-Curve	Переобтяжений. Заповнює замкнуту фундаментальну криву, визначувану масивом структур Point
FillEllipse	Переобтяжений. Заповнює внутрішню частину еліпса, який визначається обмежуючим прямокутником, заданим за допомогою пари координат – ширини і висоти

Продовження табл. 1.2

FillPath	Заповнює внутрішню частину об'єкта GraphicsPath
FillPie	Переобтяжений. Заповнює внутрішню частину сектора, визначеного еліпсом, який заданий шириною, висотою і двома радіальними лініями
FillPolygon	Переобтяжений. Заповнює внутрішню частину багатокутника, визначеного масивом крапок, заданих структурами Point
FillRectangle	Переобтяжений. Заповнює внутрішню частину прямокутника, який визначений парою координат, шириною і висотою
FillRectangles	Переобтяжений. Заповнює внутрішню частину набору прямокутників, визначуваного структурами Rectangle
FillRegion	Заповнює внутрішню частину об'єкта Region
Flush	Переобтяжений. Викликає примусове виконання всіх відкладених графічних операцій і негайно повертається, не чекаючи їх закінчення
FromHdc	Статичний. Переобтяжений. Створює об'єкт Graphics із указаного дескриптора для контексту пристрою
FromHdcInternal	Статичний. Внутрішній метод. Не використовується
FromHwnd	Статичний. Створює новий об'єкт Graphics з указаного дескриптора для вікна
FromHwnd-Internal	Статичний. Внутрішній метод. Не використовується
FromImage	Статичний. Створює новий об'єкт Graphics із заданого об'єкта Image
GetHalftone-Palette	Статичний. Отримує дескриптор потокової півтонової палітри Windows
GetHashCode	Служить хэш-функцією для конкретного типу
GetHdc	Отримує дескриптор контексту пристрою, пов'язаний з даним об'єктом Graphics
GetLifetime-Service	Витягує службовий об'єкт потокового терміну дії, який управляє засобами терміну дії даного екземпляра
GetNearest-Color	Отримує колір, найближчий до указаної структури Color
GetType	Повертає Type потокового екземпляра

Продовження табл. 1.2

InitializeLife-timeService	Отримує службовий об'єкт терміну дії, для управління засобами терміну дії даного екземпляра
IntersectClip	Переобтяжений. Оновлює вирізану область об'єкта Graphics, включаючи в неї перетин потокової вирізаної області і вказаної структури Rectangle
IsVisible	Переобтяжений. Указує, чи міститься крапка, що задана за допомогою пари координат, у видимій вирізаній області даного об'єкта Graphics
Measure-CharacterRanges	Отримує масив об'єктів Region, кожен з яких зв'язує діапазон позицій символів у рамках указаного рядка
MeasureString	Переобтяжений. Вимірює вказаний рядок у процесі її створення за допомогою заданого об'єкта Font
Multiply-Transform	Переобтяжений. Помножує універсальне перетворення даного об'єкта Graphics на перетворення вказаного об'єкта Matrix
ReleaseHdc	Звільняє дескриптор контексту пристрою, отриманий у результаті попереднього виклику методу GetHdc даного об'єкта Graphics
ReleaseHdc-Internal	Внутрішній метод. Не використовується
ResetClip	Скидає вирізану область даного об'єкта Graphics і робить її нескінченною
ResetTransform	Скидає матрицю універсального перетворення даного об'єкта Graphics і робить її одиничною матрицею
Restore	Відновлює стан даного об'єкта Graphics, повертаючи його до стану об'єкта GraphicsState
RotateTransform	Переобтяжений. Застосовує задане обертання до матриці перетворення даного об'єкта Graphics
Save	Зберігає потоковий стан даного об'єкта Graphics і пов'язує збережений стан з об'єктом GraphicsState
ScaleTransform	Переобтяжений. Застосовує вказану операцію масштабування до матриці перетворення даного об'єкта Graphics шляхом її додавання до матриці перетворення об'єкта
SetClip	Переобтяжений. Задає як вирізану область даного об'єкта Graphics властивість Clip указаного об'єкта Graphics

Закінчення табл. 1.2

ToString	Повертає String, який показує потоковий Object
TransformPoints	Переобтяжений. Перетворить масив крапок з одного координатного простору в інше, використовуючи потокове універсальне перетворення і перетворення сторінки об'єкта Graphics
TranslateClip	Переобтяжений. Переводить вирізану область об'єкта Graphics в указаному об'ємі в горизонтальному і вертикальному напрямах
TranslateTransform	Переобтяжений. Додає заданий переклад до матриці перетворення даного об'єкта Graphics
Захищені методи	
Finalize	Перевизначений. Див. Object.Finalize. У мовах C# и C++ для функцій фіналізації використовується синтаксис деструкції
MemberwiseClone	Створює неповну копію потокового Object

1.5. Клас GraphicsPath

GraphicsPath – клас, що надає послідовність сполучених ліній і кривих. Застосування використовують контури для відображення контурів фігур, заповнення внутрішніх областей фігур, створення зон відсікань. Контур може складатися з будь-якої кількості фігур (контурів). Кожна фігура або складена з послідовності ліній і кривих, або є геометричним примітивом. Початкова точка фігури – перша крапка в послідовності сполучених ліній і кривих. Кінцева крапка – остання крапка в послідовності.

Фігура, що складається з послідовності сполучених ліній і кривих (початкова і кінцева точки можуть співпадати), є розімкненою фігурою, якщо вона не замкнута явно. Фігура може бути замкнута явно за допомогою методу CloseFigure, який замикає фігуру шляхом з'єднання кінцевої і початкової крапок лінією. Фігура, що складається з геометричного примітиву, є замкнутою.

Для цілей заповнення та відсікання (наприклад, якщо контур візуалізується за допомогою методу Graphics.FillPath) всі розімкнені фігури замикаються шляхом додавання лінії від першої точки фігури до останньої. Нова фігура починається неявно, коли створюється контур або фігура замикається. Нова фігура створюється явно, коли викликається метод StartFigure.

При додаванні до контуру геометричного примітива виконується додавання фігури, що містить геометричний примітив, а також неявно починається нова фігура. Отже, в контурі завжди існує потокова фігура. Коли лінії і криві додаються до контуру, то додається неявна лінія, що сполучає кінцеву точку потокової фігури з початковою точкою нових ліній і кривих, щоб сформувати послідовність сполучених ліній і кривих.

У фігури є напрям, що визначає, як відрізки прямих і кривих прямують від початкової точки до кінцевої. Напрям задається або порядком додавання ліній і кривих до фігури, або геометричним примітивом. Напрям використовується для визначення внутрішніх зон контуру для цілей відсікання і заповнення. Список членів класу наведено в табл. 1.3.

Таблиця 1.3 – Члени класу GraphicsPath

Список членів класу GraphicsPath	Призначення
Конструктор	
GraphicsPath	Ініціалізує новий екземпляр класу GraphicsPath з перерахуванням FillMode із Alternate
Властивості	
FillMode	Отримує або задає перерахування FillMode, що визначає, як заповнюються внутрішні області фігур в об'єкті GraphicsPath
PathPoints	Отримує точки в контурі
PathTypes	Отримує типи відповідних точок в масиві PathPoints
PointCount	Отримує кількість елементів у масиві PathPoints або PathTypes

Методи класу GraphicsPath наведені в табл. 1.4.

Таблиця 1.4 – Методи класу GraphicsPath

Методи класу GraphicsPath	Призначення
AddArc	Приєднує дугу еліпса до потокової фігури
AddBezier	Додає в потокову фігуру криву Безье третього порядку
AddBeziers	Додає в потокову фігуру послідовність сполучених кривих Безье третього порядку

Продовження табл. 1.4

AddClosedCurve	Додає замкнуту криву до даного контуру. Використовується крива фундаментального сплайна, оскільки крива проходить через всі точки масиву
AddCurve	Додає в потокову фігуру криву сплайна. Використовується крива фундаментального сплайна, оскільки крива проходить через всі точки масиву
AddEllipse	Додає еліпс до потокового шляху
AddLine	Додає відрізок прямої до об'єкта GraphicsPath
AddLines	Додає послідовність сполучених відрізків прямих у кінець об'єкта GraphicsPath
AddPath	Додає вказаний об'єкт GraphicsPath до даного контуру
AddPie	Додає контур сектора до даного контуру
AddPolygon	Додає багатокутник до даного контуру
AddRectangle	Додає прямокутник до даного контуру
AddRectangles	Додає послідовність прямокутників
AddString	Додає рядок тексту в даний шлях
ClearMarkers	Видаляє всі маркери з даного контуру
Clone	Створює точну копію даного контуру
CloseAllFigures	Замикає всі незамкнуті фігури в даному контурі і відкриває нову фігуру. Кожна незамкнута фігура замикається шляхом з'єднання її початкової і кінцевої точок лінією
CloseFigure	Замикає потокову фігуру і відкриває нову. Якщо потокова фігура містить послідовність сполучених ліній і кривих, то метод замикає її шляхом з'єднання початкової і кінцевої точок
CreateObjRef (успадковано від Marshal By RefObject)	Створює об'єкт, який містить всю необхідну інформацію для конструювання прокси-серверу, використовуваного для комунікації з видаленими об'єктами
Dispose	Звільняє всі ресурси, використовувані об'єктом GraphicsPath
Equals (успадковано від Object)	Визначає, чи рівні два екземпляри Object

Продовження табл. 1.4

Flatten	Перетворить кожну криву в даному контурі в послідовність сполучених відрізків прямих
GetBounds	Повертає прямокутник, що обмежує об'єкт GraphicsPath
GetHashCode (успадковано від Object)	Служить хэш-функцією для конкретного типу, придатний для використання в алгоритмах хешування і структурах даних, наприклад, у хєш-таблиці
GetLastPoint	Отримує останню точку масиву PathPoints объекта GraphicsPath
GetLifetimeService (успадковано від MarshalByRefObject)	Витягує службовий об'єкт потокового терміну дії, який управляє засобами терміну дії даного екземпляра
GetType (успадковано від Object)	Повертає Type потокового екземпляра
InitializeLifetimeService (успадковано від MarshalByRefObject)	Отримує службовий об'єкт, для управління засобами терміну дії даного екземпляра
IsOutlineVisible	Указує, чи міститься певна точка всередині (під) контуру об'єкта GraphicsPath при відображенні його за допомогою вказаного об'єкта Pen
IsVisible	Визначає, чи міститься вказана точка в об'єкті GraphicsPath
Reset	Очищає масиви PathPoints и PathTypes і встановлює FillMode в Alternate
Reverse	Змінює порядок точок у масиві PathPoints об'єкта GraphicsPath на протилежний
SetMarkers	Установлює маркер на об'єкті GraphicsPath
StartFigure	Відкриває нову фігуру, не замикаючи при цьому потокову. Всі подальші точки, що додаються до контуру, додаються до нової фігури
ToString (успадковано від Object)	Повертає String, який подає потоковий Object
Transform	Застосовує матрицю перетворення до об'єкта GraphicsPath

Закінчення табл. 1.4

Warp	Застосовує перетворення перекосу, визначуване прямокутником і паралелограмом, до об'єкта GraphicsPath
Захищені методи	
Finalize	Перевизначений. Див. Object.Finalize.
MemberwiseClone (успадковано від Object)	Створює неповну копію потокового Object

1.6. Клас Region

Клас Region (Область) призначається для створення об'єктів, які описують внутрішню частину графічної форми з прямокутників і фігур, складених із замкнутих ліній. Цей клас не успадковується.

Область є такою, що масштабується. Додаток може використовувати області для фіксації вихідних даних операцій рисування. Диспетчер вікон застосовує області для визначення області зображення вікон. Ці області називаються вирізаними. Додаток може також використовувати області в операціях перевірки наявності даних, наприклад, перетини точки або прямокутника з областю. Додаток може заповнювати область за допомогою об'єкта Brush.

Безліч пікселів, що входять до складу регіону, може складатися з декількох несуміжних ділянок.

Методи класу наведені в табл. 1.5–1.6.

Таблиця 1.5 – Відкриті методи класу Region

Методи класу Region	Призначення
Clone	Створює точну копію об'єкта Region
Complement	Оновлює об'єкт Region, щоб включити частину вказаної структури RectangleF, не пересічну з об'єктом Region
CreateObjRef (унаследовано от MarshalByRefObject)	Створює об'єкт, який містить всю необхідну інформацію для створення проксі-серверу, використовуваного для комунікації з видаленими об'єктами
DisposeEquals	Звільняє всі ресурси, використовувані об'єктом Region
Exclude	Оновлює об'єкт Region, щоб включити частину його внутрішньої частини, не пересічну з указаною структурою Rectangle

Продовження табл. 1.5

FromHrgn	Ініціалізував новий об'єкт Region з дескриптора вказаної існуючої області GDI
GetBounds	Повертає структуру RectangleF, яка подає прямокутник, що обмежує об'єкт Region на поверхні малюнка об'єкту Graphics
GetHashCode (успадковано від Object)	Служить хеш-функцією для конкретного типу, придатний для використання в алгоритмах хешування і структурах даних
GetHrgn	Повертає дескриптор Windows для об'єкта Region в указаному графічному контексті
GetLifetimeService (успадковано від MarshalByRefObject)	Витягує службовий об'єкт потокового терміну дії, який управляє засобами терміну дії даного екземпляра
GetRegionData	Повертає об'єкт RegionData, який подає дані, що описують об'єкт Region
GetRegionScans	Повертає масив структур RectangleF, що апроксимують об'єкт Region
GetType (успадковано від Object)	Повертає Type потокового екземпляра
InitializeLifetimeService (успадковано від MarshalByRefObject)	Отримує службовий об'єкт терміну дії для управління засобами терміну дії даного екземпляра
Intersect	Замінює об'єкт Region на його перетин з указаним об'єктом Region
IsEmpty	Перевіряє чи має об'єкт Region порожню внутрішню частину на вказаній поверхні рисунка
IsInfinite	Перевіряє чи має об'єкт Region порожню внутрішню частину на вказаній поверхні рисунка
IsVisible	Перевіряє чи міститься вказаний прямокутник в об'єкті Region
MakeEmpty	Ініціалізував об'єкт Region для порожньої внутрішньої частини
MakeInfinite	Ініціалізував об'єкт Region для нескінченної внутрішньої частини
ToString (успадковано від Object)	Повертає String, який подає потоковий Object

Закінчення табл. 1.5

Transform	Перетворить цей об'єкт Region за допомогою вказаного об'єкта Matrix
Translate	Зміщує координати об'єкта Region на вказану величину
Union	Замінює об'єкт Region на його об'єднання з указаним об'єктом GraphicsPath
Xor	Замінює об'єкт Region на різницю об'єднання і його перетину з указаним об'єктом GraphicsPath

Таблиця 1.6 – Захищені методи класу Region

Методи класу Region	Призначення
Finalize	Перевизначений. Див. Object.Finalize. У мовах C# и C++ для функцій фіналізації використовується синтаксис деструкції
MemberwiseClone (успадковано від Object)	Створює неповну копію потокового Object

1.7. Власні елементи управління

Відомі підходи до розробки нових елементів управління:

– об'єднання стандартних елементів управління в групи (складові елементи управління);

– оголошення нових класів, що успадковують від існуючих елементів управління;

– написання нових елементів "з нуля".

Розробка складових елементів управління припускає оголошення класу, похідного від класу UserControl і використання Майстра UserControl, для додавання вкладених елементів управління з подальшою настройкою створюючих елементів.

Новий елемент управління може бути побудований на основі класу – спадкоємця якого-небудь із існуючих елементів управління. У цьому випадку в новому класі вдається частково використовувати функціональності раніше оголошеного класу, можливо, зберігаючи при цьому зовнішній вигляд елемента. Наприклад, можна оголосити власний варіант класу кнопки, який успадковуватиме клас Button.

Написання нового елемента "з нуля" відрізняється від попереднього варіанта розробки вибором базового класу. У цьому випадку ґрунтуються на класі Control, який не надає нащадкам навіть елементарного графічного інтерфейсу. Процес візуалізації в цьому випадку

забезпечується обробником події Paint, що перевизначається. При цьому перевизначається віртуальний метод базового класу OnPaint з єдиним аргументом типу PaintEventArgs, який містить інформацію про клієнтську область елемента управління. Член цього класу – об'єкт типу Graphics – забезпечує формування подання елемента управління. Другий член класу – об'єкт типу ClipRectangle – описує доступну клієнтську область елемента управління.

Слід зазначити, що між двома останніми способами визначення елементів управління не існує чітких меж. В обох випадках підставою для класифікації виявляється об'єм роботи з довизначення і перевизначення методів і властивостей новостворюваного класу елементів управління.

Об'єкти класу ImageList призначаються для збереження рисунків, які можуть відображатися іншими елементами управління. У загальному випадку цей компонент дозволяє написати код для уніфікованого каталога рисунків. До кожного рисунка можна дістати доступ за допомогою значення індексу цього рисунка. Рисунки, що відображаються, мають один і той же формат і розмір, що встановлюється у властивості ImageSize. Таким чином, на основі даної властивості може бути реалізований ефект масштабування елемента управління в сенсі зміни його видимих розмірів у разі зміни клієнтських розмірів форми.

1.8. Створення додатків

Лістинг 1.1. Програма рисування лінії, прямокутника та еліпса:

```
title рисування лінії, прямокутника та еліпса
.686                ; директива визначення типу мікропроцесора
.model flat,stdcall ; завдання лінійної моделі пам’яті та угоди ОС Windows
option casemap:none             ; відмінність малих та великих літер
include \masm32\include\windows.inc ; файли структур, констант …
include \masm32\macros\macros.asm
uselib user32,gdiplus,kernel32,gdi32
  GdiplusStartupInput STRUCT
    GdiplusVersion DWORD ?
    DebugEventCallback DWORD ?
    SuppressBackgroundThread DWORD ?
    SuppressExternalCodecs DWORD ?
  GdiplusStartupInput ENDS

  UnitPixel equ 2 ; const
.data
      szAppName db "Work with GDI+",0
      szClassName db "GdipExample",0
```

```
        rWidth REAL4 15.0
    rX1 REAL4 10.0      ; X-координата початку лінії
    rY1 REAL4 10.0      ; Y-координата початку лінії
    rX2 REAL4 450.0     ; X-координата закінчення лінії
    rY2 REAL4 300.0     ; Y-координата закінчення лінії
    .data?
        hInstance dd ?
        hWndMain dd ?
.code
WinMain proc  hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,\
                                                CmdShow:DWORD
        LOCAL wc:WNDCLASSEX
        LOCAL msg:MSG
        LOCAL gdiplusStartupInput:GdiplusStartupInput
        LOCAL gdiplusToken:DWORD
mov gdiplusStartupInput.GdiplusVersion,1 ; ініціалізація бібліотеки
and gdiplusStartupInput.DebugEventCallback,0
and gdiplusStartupInput.SuppressBackgroundThread,0
and gdiplusStartupInput.SuppressExternalCodecs,0
invoke GdiplusStartup,addr gdiplusToken,addr gdiplusStartupInput,NULL
        mov wc.cbSize,sizeof WNDCLASSEX
        mov wc.style,CS_HREDRAW or CS_VREDRAW
        mov wc.lpfnWndProc,offset WndProc
        mov wc.cbClsExtra,NULL
        mov wc.cbWndExtra,NULL
        push hInst
        pop wc.hInstance
        mov wc.hbrBackground,COLOR_WINDOW+1
        mov wc.lpszMenuName,NULL
        mov wc.lpszClassName,offset szClassName
        invoke LoadIcon,NULL,IDI_APPLICATION
        mov wc.hIcon,eax
        mov wc.hIconSm,eax
        invoke LoadCursor,NULL,IDC_ARROW
        mov wc.hCursor,eax
invoke RegisterClassEx,addr wc
invoke CreateWindowEx,WS_EX_APPWINDOW,addr szClassName, \
  addr szAppName, WS_TILEDWINDOW,\
    CW_USEDEFAULT,CW_USEDEFAULT, 500, 350, 0, 0, hInst, 0
        mov hWndMain,eax
        invoke ShowWindow,hWndMain,SW_SHOWNORMAL
        invoke UpdateWindow,hWndMain
        .while TRUE
                invoke GetMessage,addr msg,0,0,0
                .break .if (!eax)
                invoke TranslateMessage,addr msg
```

```
                invoke DispatchMessage,addr msg
        .endw
invoke GdiplusShutdown,addr gdiplusToken ; завершення роботи з бібл.
        mov eax,msg.wParam
        ret
    WinMain endp

WndProc proc hWnd:DWORD,uMsg:DWORD,wParam:DWORD,\
            lParam:DWORD
        LOCAL hdc:DWORD     ; резервування стека під контекст
        LOCAL ps:PAINTSTRUCT
        LOCAL graphics:DWORD
        LOCAL pen:DWORD
                .if uMsg==WM_CREATE
                ;push hWnd
                ;pop hWndMain
.elseif uMsg==WM_PAINT  ; якщо є повідомлення про перерисовування
    invoke BeginPaint,hWnd,addr ps ; виклик підготовчої процедури
        mov hdc,eax             ; збереження контексту
invoke GdipCreateFromHDC,hdc,addr graphics; створення контексту
    invoke GdipCreatePen1,05F0000FFh,\ ; 7F – прозорість, RGB
      rWidth,UnitPixel,addr pen
    invoke GdipDrawLine,graphics,pen,rX1,rY1,rX2,rY2
    invoke GdipDrawRectangle,graphics,pen,rX1,rY1,rX2,rY2
    invoke GdipDrawEllipse,graphics,pen,rX1,rY1,rX2,rY2

        invoke GdipDeletePen,pen
        invoke GdipDeleteGraphics,graphics
        invoke EndPaint,hWnd,addr ps
.elseif uMsg==WM_DESTROY ; якщо є повідомлення про знищення вікна
invoke PostQuitMessage,NULL ; передача повідомлення про знищення
        .else
        invoke DefWindowProc,hWnd,uMsg,wParam,lParam
        ret
        .endif          ; закінчення логічної структури .IF – .ELSEIF
        xor eax,eax
        ret             ; повернення з процедури
    WndProc endp        ; закінчення процедури
start:
    invoke  GetModuleHandle, NULL; отримання дескриптора програми
     mov   hInstance,eax            ; збереження дескриптора програми
    invoke WinMain,hInstance,NULL,NULL,SW_SHOWDEFAULT
invoke ExitProcess,0 ; повернення управління та вивільнення ресурсів
    end start
```

Результат виконання програми наведено на рисунку 1.1.

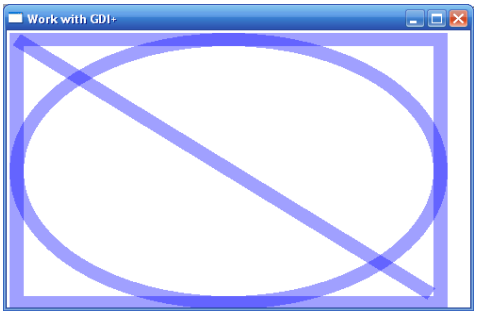


Рисунок 1.1 – Результат роботи програми з лістингу 1.1

Лістинг 1.2. Програма рисування 18-кутової фігури з точками на колі:

```
.386
.model flat,stdcall
option casemap:none
WinMain proto :DWORD,:DWORD,:DWORD,:DWORD
include \masm32\include\windows.inc
include \masm32\macros\macros.asm
uselib user32,gdiplus,kernel32,gdi32
    GdiplusStartupInput STRUCT
      GdiplusVersion DWORD ?
      DebugEventCallback DWORD ?
      SuppressBackgroundThread DWORD ?
      SuppressExternalCodecs DWORD ?
    GdiplusStartupInput ENDS
.const
    IDM_EXIT equ 2
    IDM_ABOUT equ 3
    IDI_ICON  equ 22
    UnitPixel equ 2
.data
    ClassName db "SimpleWinClass",0
    AppName  db "Курсовой с использованием GDI+ Double buffer",0
    MenuName db "FirstMenu",0
About_string db "Програму написал студент гр. КИТ-30 Поддубный Д.В.",0
    wTop dd ?                    ; відступ зверху
    wLeft dd ?                   ; відступ знизу
    wH dd 500                    ; висота вікна
    wW dd 600                    ; ширина вікна
    cRect RECT <>                ; область вікна
    ;GDI+
```

```
        memGraphics             dd 0
        hBitmap         dd 0
        rWidth dd 1.0           ; ширина щітки в пікселях
        Yc tbyte ?              ; координати відносно центра
        Xc tbyte ?              ; координати відносно центра
        cR tbyte 200.0          ; радіус
        dPoints_count dd 18  ; кількість точок
cColor dd 0FF0000FFh  ; перший байт повинен бути FF, останні 3
описують R,G,B
            tmp dd ?                    ; тимчасова змінна
            x2 dword 2                  ; x*2
            dPoints qword 100 dup(0)     ; масив точок (max = 100)
pushre macro
            push eax
            push ebx
            push ecx
            push edx
            push esi
            push edi
        endm
popre macro
            pop edi
            pop esi
            pop edx
            pop ecx
            pop ebx
            pop eax
        endm
.data?
            hInstance HINSTANCE ?
            CommandLine LPSTR ?
.code
        CalcPoints proc base_addr:dword
            LOCAL i:dword
            pushre
            ; Xi = Xc + R*cos(2*pi*i/n)
            ; Yi = Yc + R*sin(2*pi*i/n)
        finit
            mov edi,base_addr
            mov eax,dPoints_count
            inc eax
            mov ebx,1
calc_points:
            mov dword ptr[i],ebx

; X[i]
```

```
        fld tbyte ptr[Xc]
        fistp dword ptr[tmp]        ; tmp = Xc
        fild dword ptr[dPoints_count] ;st(0) = N
        fidivr dword ptr [i]        ; st(0) = index/N
        fldpi
        fmul                        ; st(0) = (index/N)*pi
        fimul dword ptr[x2]         ; st(0) = (index/N)*pi*2
        fcos                        ; st(0) = cos((index/N)*pi*2)
        fld tbyte ptr[cR]
        fmul                        ; st(0) = R*cos((index/N)*pi*2)
        fld tbyte ptr[Xc]
        fadd st(0),st(1)            ; st(0) = Xc+R*cos((index/N)*pi*2)
        ffree st(1)
        fistp dword ptr[tmp]
                                    ; put into array (first X)
        lea esi,tmp
        mov ecx,4
        rep movsb                   ; write 2x2 words into edi [base_addr]
; Y[i]
        fld tbyte ptr[Yc]
        fistp dword ptr[tmp]        ; tmp = Yc
        fild dword ptr[dPoints_count] ; st(0) = N
        fidivr dword ptr [i]        ; st(0) = index/N
        fldpi
        fmul                        ; st(0) = (index/N)*pi
        fimul dword ptr[x2]         ; st(0) = (index/N)*pi*2
        fsin                        ; st(0) = sin((index/N)*pi*2)
        fld tbyte ptr[cR]
        fmul                        ; st(0) = R*sin((index/N)*pi*2)
        fld tbyte ptr[Yc]
        fadd st(0),st(1)            ; st(0) = Yc+R*sin((index/N)*pi*2)
        ffree st(1)
        fistp dword ptr[tmp]
                                    ; put into array (second Y)
        lea esi,tmp
        mov ecx,4
        rep movsb          ; write 2x2 words into edi [base_addr]

        inc ebx
        cmp eax,ebx
        jne calc_points
        popre
        ret
    CalcPoints endp

    CalcRect proc h:HWND
```

```
            finit
            invoke  GetClientRect, h, addr cRect
            fild dword ptr[cRect.right]
            fidiv dword ptr[x2]
            fstp tbyte ptr[Xc]
            ffree st(0)
            fild dword ptr[cRect.bottom]
            fidiv dword ptr[x2]
            fstp tbyte ptr[Yc]
            ffree st(0)
            invoke  InvalidateRgn, h, NULL, FALSE
           ret
      CalcRect endp

      PaintFigureDoubleBuffered proc hwnd:DWORD
           LOCAL hdc:HDC
           LOCAL hGraphics:DWORD
           LOCAL pen:DWORD        ; дескриптор щітки
           LOCAL dWidth:DWORD
           LOCAL dX1:DWORD
           LOCAL dY1:DWORD
           LOCAL dX2:DWORD
           LOCAL dY2:DWORD
           LOCAL gdiplusStartupInput:GdiplusStartupInput
           LOCAL gdiplusToken:DWORD

           mov gdiplusStartupInput.GdiplusVersion,1
           and gdiplusStartupInput.DebugEventCallback,0
           and gdiplusStartupInput.SuppressBackgroundThread,0
           and gdiplusStartupInput.SuppressExternalCodecs,0

           invoke  GetDC, hwnd
           mov hdc, eax
invoke GdiplusStartup,addr gdiplusToken,addr gdiplusStartupInput,0
                                                              ; ініціалізація
invoke GdipCreateFromHDC,hdc,addr hGraphics   ;
invoke GdipCreateBitmapFromGraphics, dword ptr[cRect.right],dword
ptr[cRect.bottom], hGraphics,addr hBitmap              ; бітмап
invoke GdipGetImageGraphicsContext, hBitmap,addr memGraphics
invoke GdipSetSmoothingMode, memGraphics, 4        ; тип згладжування
invoke GdipGraphicsClear , memGraphics, 0ffffffffh
                                                   ; очищаємо memGraphics
      m2m dWidth,rWidth
invoke GdipCreatePen1,cColor,dWidth,UnitPixel,addr pen ; перо
           mov eax,0
           lea ecx,dPoints
```

```
            mov esi,ecx
            mov edi,esi
            .REPEAT
                        mov ebx,0
                        mov edi,ecx
                        .REPEAT
                                    .IF edi!=esi
                                                m2m dX1,[edi]
                                                m2m dY1,[edi+4]
                                                m2m dX2,[esi]
                                                m2m dY2,[esi+4]
                                                pushre
invoke GdipDrawLineI,memGraphics,pen,dX1,dY1,dX2,dY2 ; рисуємо лінію
                                                popre
                                    .ENDIF
                                    mov edx,8
                                    add edi,edx
                                    inc ebx
                        .UNTIL ebx==dPoints_count
                        mov edx,8
                        add esi,edx
                        inc eax
            .UNTIL eax==dPoints_count
invoke GdipDrawImagePointRectI, hGraphics, hBitmap, 0, 0, 0, 0, dword
ptr[cRect.right],dword ptr[cRect.bottom], 2
            invoke GdipDeletePen,pen
            invoke GdipDeleteGraphics, memGraphics
            invoke GdipDisposeImage, hBitmap
            invoke GdipDeleteGraphics, hGraphics
            invoke GdiplusShutdown,addr gdiplusToken  ; Вбиваємо GDI+
            ret
      PaintFigureDoubleBuffered endp

WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,\
             CmdShow:DWORD
            LOCAL wc:WNDCLASSEX
            LOCAL msg:MSG
            LOCAL hwnd:HWND
mov   wc.cbSize,SIZEOF WNDCLASSEX  ; кількість байтів структури
mov   wc.style, CS_HREDRAW or CS_VREDRAW ; стиль та поведінка вікна
mov   wc.lpfnWndProc, OFFSET WndProc  ; адреса процедури WndProc
mov   wc.cbClsExtra,NULL              ; кількість байтів для структури
mov   wc.cbWndExtra,NULL              ; кількість байтів для структури
      m2m hInst,wc.hInstance
      mov   wc.hbrBackground,COLOR_WINDOW+1   ; колір вікна
      mov   wc.lpszMenuName,OFFSET MenuName   ; ім'я ресурсу меню
```

```
        mov   wc.lpszClassName,OFFSET ClassName   ; ім'я класу
        invoke  LoadIcon,hInstance, IDI_ICON   ; ресурс піктограми
        mov   wc.hIcon,eax                ; дескриптор піктограми
        mov   wc.hIconSm,eax              ; дескриптор маленького віконця
        invoke LoadCursor,NULL,IDC_ARROW ; ресурс курсора
        mov   wc.hCursor,eax
invoke RegisterClassEx, addr wc     ;  реєстрація класу вікна
invoke GetSystemMetrics,SM_CXSCREEN    ;  ширина екрана в пікселях
            shr eax, 1
            mov ebx, wW
            shr ebx, 1
            sub eax, ebx
            mov wLeft, eax
invoke GetSystemMetrics,SM_CYSCREEN   ; висота екрана в пікселях
            shr eax, 1
            mov ebx, wH
            shr ebx, 1
            sub eax, ebx
            mov wTop, eax
invoke CreateWindowEx,WS_EX_LEFT, ADDR ClassName, ADDR
AppName, WS_OVERLAPPEDWINDOW, wLeft,wTop,wW,wH, 0,0, hInst,0
            mov   hwnd,eax
; розрахунок точок і розмірів центра вікна
        invoke CalcRect,hwnd
        invoke CalcPoints,addr dPoints

        invoke ShowWindow, hwnd,SW_SHOWNORMAL
        invoke UpdateWindow, hwnd
            .WHILE TRUE
                 Invoke GetMessage, ADDR msg,NULL,0,0
                .BREAK .IF (!eax)
                 invoke TranslateMessage,addr msg
                 Invoke DispatchMessage, ADDR msg
            .ENDW
            mov eax,msg.wParam
            ret
        WinMain endp

WndProc proc hWnd:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM
.IF uMsg==WM_DESTROY
                    invoke PostQuitMessage,NULL
.ELSEIF uMsg==WM_CREATE
.ELSEIF uMsg==WM_COMMAND
            mov eax,wParam
            .IF ax==IDM_ABOUT
```

```
            invoke MessageBox,NULL,addr About_string,addr
AppName,MB_OK
            .ENDIF
            .IF ax==IDM_EXIT
            invoke DestroyWindow,hWnd
            .ENDIF
.ELSEIF uMsg==WM_PAINT  ; рисуємо в повідомленні WM_PAINT
invoke PaintFigureDoubleBuffered,hWnd ; рисуємо з подвійним буфером
.ELSEIF uMsg==WM_ERASEBKGND
          ; прибираємо ефект мерехтіння, забороняючи очищення фону вікна
            mov eax,1
.ELSEIF uMsg==WM_SIZE ; перераховуємо крапки при збільшенні розміру
            invoke CalcRect,hWnd
            invoke CalcPoints,addr dPoints
.ELSE
            invoke DefWindowProc,hWnd,uMsg,wParam,lParam
            ret
.ENDIF
            xor esi,esi
            xor eax,eax
            ret
      WndProc endp

_start:
      invoke GetModuleHandle, NULL ; отримання дескриптора програми
            mov hInstance,eax            ; збереження дескриптора програми
            invoke GetCommandLine
            mov CommandLine,eax
      invoke WinMain, hInstance,NULL,CommandLine, SW_SHOWDEFAULT
      invoke ExitProcess,eax
      end _start
```

Результат виконання програми наведено на рисунку 1.2.

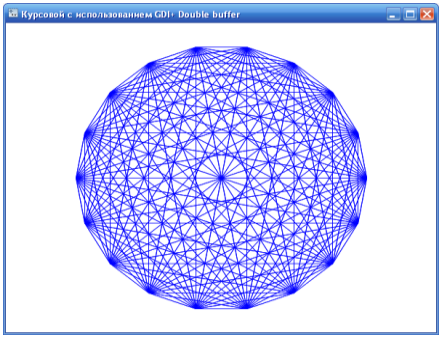


Рисунок 1.2 – 18-кутова фігура з точками на колі

2. НЕСТАНДАРТНІ ВІКНА

2.1. Основні функції для роботи з нестандартними вікнами

Windows API надає набір функцій, що дозволяють описати довільну геометричну фігуру, яка потім може використовуватися при роботі з вікнами. Такими функціями є регіони.

Регіон можна подати у вигляді поверхні, обмеженої координатами з кутовими точками цієї області. Таким чином, можна описати область будь-якої форми, потім створити з неї за допомогою спеціальних функцій регіон і "прикріпити" його до потрібного вікна, наприклад так, як наведено за адресами http://www.manhunter.ru/ assembler, http://www.rsdn.ru/article/gdi/ regions. xml, http://cppbuilder.ru/ articles/0033.php.

У Microsoft Windows регіоном називається прямокутник, полігон або еліпс (або комбінація два або більш за ці фігури), які можуть бути заповнені, нарисовані, інвертовані, обрамлені і використовуються для визначення місцезнаходження курсора (мається на увазі, що є стандартна функція, яка визначає: чи входить крапка (*X*, *Y*) в створений регіон).

Регіони можуть комбінуватися між собою із застосуванням логічних операцій OR, XOR й т.ін.

Регіони використовують:

– для створення різних форм вікон і областей рисування в різні кольори;

– для обмеження області рисування, додаючи їм різні форми;

– для визначення потрапляння на об'єкт на екрані;

– для відділення заголовка або меню від основного вікна;

– створення маски для обмеження рисування (clipping).

Для роботи з регіонами передбачено дві функції:

SetWindowRgn(HRGN hRgn, BOOL bRedraw); // встановити регіон;
GetWindowRgn(HRGN hRgn)const; // отримати регіон.

У функції SetWindowRgn є ряд особливостей:

– координати регіону повинні задаватися відносно верхнього лівого кута вікна, а не відносно клієнтської області або верхнього лівого кута екрана;

– функція SetWindowRgn передає системі управління регіоном за допомогою хендла. При цьому система не робить копії об'єкта, а це означає, що можна не модифікувати об'єкт і не видаляти його;

– SetWindowRgn не працює для дочірніх вікон – вікон, створених із стилем WS_CHILD.

Для того щоб використовувати регіон його потрібно створити. Для створення регіонів використовуються функції:

CreateRectRgn(int x1, int y1, int x2, int y2); // прямокутний регіон;

CreateRectRgnIndirect(LPCRECT lpRect); // прямокутний регіон із структурою Rect.

Функція **CreateRectRgn** створює регіон прямокутної форми (параметри *X, Y* задають для верхнього лівого і нижнього правого кута прямокутника регіону).

Функція **CreateRectRgnIndirect** аналогічна функції CreateRectRgn – різниця в тому, що замість чотирьох координат функція отримує структуру типу Rect.

CreateRoundRectRgn(int x1, int y1, int x2, int y2, int x3, int y3);// регіон із закругленими краями.

Функція CreateRoundRectRgn аналогічна функції CreateRectRgn, але прямокутна область має кути, що округляють. *x*1, *y*1 – координати верхнього лівого кута, *x*2, *y*2 – координати нижнього правого, *x*3, *y*3 – висота і ширина еліпса, вживаного для округлення кутів.

CreatePolygonRgn(LPPOINT lpPoints, int nCount, int nMode); // регіон із масиву крапок.

Функція CreatePolygonRgn створює регіон, обмежений багатокутником, вершини якого задані в масиві Points (елемент масиву – пара координат *x, y*). Параметр Count указує кількість крапок у масиві nMode та визначає, які крапки належатимуть регіону (можливі два режими – ALTERNATE та WINDING).

CreatePolyPolygonRgn(LPPOINT lpPoints, LPINT lpPolyCounts, int nCount, int nPolyFillMode); // регіон з набору регіонів.

Функція CreatePolyPolygonRgn створює регіон, що складається з декількох регіонів. Параметр lpPoints є масивом масивів крапок (двовимірний масив). Параметр lpPolyCounts є масивом, що містить кількість крапок (вершин) для кожного з регіонів nCount – кількість регіонів nPolyFillMode – режим заповнення.

CreateEllipticRgn(int x1, int y1, int x2, int y2); // еліпсоїдний регіон, де *x*1 та *y*1 – це координати верхнього лівого рівня прямокутника, куди вписано еліпс, а *x*2 та *y*2 – це координати правого нижнього кута прямокутника. Наприклад:

```
  invoke CreateEllipticRgn,200,80,440,240 ; створення еліпса 1
    mov hRgn1, eax;
  invoke CreateEllipticRgn,438,80,680,240 ; створення еліпса 2
    mov hRgn2,eax;
invoke CombineRgn,hRgn1,hRgn1,hRgn2,RGN_OR; накладення двох еліпсів
```

CreateEllipticRgnIndirect(LPCRECT lpRect); // еліпсоїдний регіон, заданий структурою Rect.

Дані функції створюють еліптичний регіон (еліпс уписаний в прямокутник, заданий у першому випадку координатами верхнього лівого і нижнього правого кутів, у другому – структурою TRect з тими ж координатами).

Регіони можна комбінувати і отримувати нові форми за допомогою функції:

CombineRgn(DestRgn, SrcRgn1, SrcRgn2: HRgn; CombineMode: Integer): Integer; // комбінування регіонів. Об'єднує області SrcRgn1 та SrcRgn2 і розміщує результат в DestRgn.

Параметр DestRgn – область приймання нового регіону.

Параметр SrcRgn1, SrcRgn2 – перший та другий регіони.

Параметр CombineMode визначає метод об'єднання областей:

RGN_AND – результуючий регіон буде перетинанням регіонів 1 і 2;

RGN_COPY – результуючий регіон буде копією регіону 1;

RGN_DIFF – результуючий регіон буде фрагментом регіону 1, що не належить регіону 2 (неналежна частина);

RGN_OR – об'єднання двох регіонів;

RGN_XOR – об'єднання регіонів без перетинання.

Функція **EqualRgn** проводить порівняння регіонів 1 і 2 і повертає true, якщо регіони 1 і 2 рівні.

Функції, які визначають потрапляння крапки всередину регіону такі:

PtInRegion(int x, int y) const;

PtInRegion(POINT point) const.

Вікна нестандартної форми за допомогою регіонів складаються з чітких геометричних форм. Проте особливий інтерес мають вікна, які подані у вигляді картинок.

Розглянемо лістинг 2.1 створення вікна у вигляді еліпса.

Лістинг 2.1. Програма створення вікна у вигляді еліпса:

```
      .486        ; директива визначення типу мікропроцесора
.model flat,stdcall    ; завдання лінійної моделі пам'яті та угоди ОС
Windows
option casemap:none          ; відмінність малих та великих літер
  include \masm32\include\windows.inc ; файли структур, констант …
  include \masm32\macros\macros.asm
  uselib user32, kernel32, gdi32
 WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD
 WndProc PROTO :DWORD,:DWORD,:DWORD,:DWORD
    @ MACRO b0,b1,b2,b3
       b0
        b1
         b2
          b3
       ENDM
.data
       szClassName db "MyClass"
       szUntitled db " "
       winWid dd 500
       winHgt dd 500
.data?
       hInstance dd ?
       CommandLine dd ?
       hWnd    dd ?
       left    dd ?
       top     dd ?
.code
 start:
     invoke GetModuleHandle, NULL ; отримання дескриптора програми
     mov    hInstance,eax         ; збереження дескриптора програми
     invoke GetCommandLine
     mov CommandLine,eax
     invoke WinMain, hInstance,NULL,CommandLine, SW_SHOWDEFAULT
    invoke ExitProcess,eax

WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,\
                                                    CmdShow:DWORD
       LOCAL wc    :WNDCLASSEX
       LOCAL msg   :MSG
```

```
@<mov wc.cbSize, sizeof WNDCLASSEX>,<mov wc.lpfnWndProc, OFFSET
WndProc>
@<mov wc.style, CS_HREDRAW or CS_VREDRAW>,< mov
wc.cbClsExtra,0>
@<mov wc.cbWndExtra,0>,< mov eax, hInst>,< mov wc.hInstance, eax>
@<mov wc.hbrBackground,  COLOR_BTNFACE+1>,<mov
wc.lpszMenuName,   0>
@<mov wc.lpszClassName,  offset szClassName>,<invoke LoadIcon,hInst,0>
mov wc.hIcon, eax                       ; дескриптор піктограми
invoke LoadCursor, 0, IDC_ARROW  ; курсор стандартний
@<mov wc.hCursor, eax>,< mov wc.hIconSm, 0>
invoke RegisterClassEx, ADDR wc
                        ; розрахунок середени екрана
invoke GetSystemMetrics,SM_CXSCREEN ;  ширина екрана в пікселях
      shr eax, 1 ; ділення на 2 – розрахунок середини по X-координаті
      mov ebx, winWid          ; X-координата регіону
      shr ebx, 1          ; розрахунок середини регіону по X-координаті
      sub eax, ebx      ; розрахунок середини екрану по X-координаті
      mov left, eax ; збереження X-координати  в комірці головного вікна
invoke GetSystemMetrics,SM_CYSCREEN ; висота екрана в пікселях
@< shr eax, 1>,< mov ebx, winHgt>,< shr ebx, 1>,< sub eax, ebx>
       mov top, eax ; збереження Y-координати  в комірці головного вікна
invoke CreateWindowEx,WS_EX_LEFT, ADDR szClassName, \
 ADDR szUntitled, WS_OVERLAPPEDWINDOW,
 left,top,winWid,winHgt, \; гориз. та верт. коорд., ширина й висота вікна
  0,0, hInst,NULL
       mov   hWnd,eax
invoke ShowWindow,hWnd,SW_SHOWNORMAL ; стан показу вікна
invoke UpdateWindow,hWnd       ; відновлення вікна через WM_PAINT
.WHILE TRUE
INVOKE GetMessage, ADDR msg,0,0,0 ; витягання повідомлення з черги
        .BREAK .IF (!eax)
INVOKE DispatchMessage, ADDR msg ; відправка на обслуговування
    .ENDW
     mov     eax,msg.wParam
ret                             ; повернення з процедури WinMain
WinMain endp                    ; закінчення процедури з ім'ям WinMain

WndProc proc hWin:DWORD, uMsg:DWORD, wParam:DWORD, \
           lParam :DWORD
      .if uMsg == WM_CREATE ; обробка  повідомлення WM_CREATE
invoke CreateEllipticRgn, 80, 0,\ ; верхній лівий кут прямокутника
    300, 200  ; правий нижній кут прямокутника
                        ; прикріплення регіону до вікна
invoke SetWindowRgn, hWin,\ ; дескриптор вікна,яке буде змінюватися
    eax, 1  ; дескриптор регіону та  перерисовування
```

```
    .elseif uMsg == WM_DESTROY ; обробка  повідомлення при знищенні
invoke PostQuitMessage, NULL ; відправлення повідомл. про знищення
        ret
    .endif
invoke DefWindowProc,hWin, uMsg, wParam, lParam
                                        ; відправка повідомлення  до WndProc
        ret                             ; повернення з процедури
        WndProc endp                    ; закінчення процедури WndProc
        end start                       ; закінчення програми з ім'ям start
```

Для того щоб вікно розмістилося точно у середині екрана виконуються розрахунки координат у фрагмент коду:

```
invoke GetSystemMetrics,SM_CXSCREEN ;  ширина екрана в пікселях
        shr eax, 1 ; ділення на 2 – розрахунок середини по X-координаті
        mov ebx, winWid             ; X-координата регіону
        shr ebx, 1          ; розрахунок середини регіону по X-координаті
        sub eax, ebx      ; розрахунок середини екрана по X-координаті
        mov left, eax ; збереження X-координати  в комірці головного вікна
invoke GetSystemMetrics,SM_CYSCREEN ; висота екрана в пікселях
        shr eax, 1 ; ділення на 2 – розрахунок середини по Y-координаті
        mov ebx, winHgt             ; Y-координата регіону
        shr ebx, 1          ; розрахунок середини регіону по Y-координаті
        sub eax, ebx      ; розрахунок середини екрана по Y-координаті
```

Результати роботи програми наведено на рисунку 2.1.

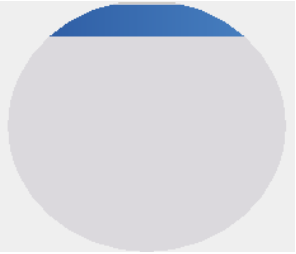


Рисунок 2.1 – Відображення вікна у вигляді еліпса

У програмі спочатку створюється головне вікно, а вже на це вікно накладується регіон. Для цього створюється регіон у вигляді еліпса функцією **CreateEllipticRgn,** а прикріпляється до вікна функцією SetWindowRgn. Якщо цю функцію закоментувати, то регіон буде створений та відобразиться у повноцінному вікні. Функція SetWindowRgn замість головного вікна відобразить тільки регіон.

Розглянемо лістинг 2.2 створення вікна у вигляді 5-кутової зірки.

Лістинг 2.2. Програма створення вікна у вигляді 5-кутової зірки:

```
        .686                    ; директива визначення типу мікропроцесора
.model flat,stdcall ; завдання лінійної моделі пам'яті та угоди ОС Windows
        option casemap:none     ; відмінність малих та великих літер
        include \masm32\include\windows.inc ; файли структур, констант …
        include \masm32\macros\macros.asm
        uselib user32,kernel32,gdi32
 WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD
 WndProc PROTO :DWORD,:DWORD,:DWORD,:DWORD
@ MACRO c0,c1,c2,c3,c4 ; макрос розміщення одним рядком п'яти
команд
    c0
     c1
      c2
       c3
        c4
    ENDM
.data
        szClassName db "MyClass"
        szUntitled db " "
        winWid dd 500
        winHgt dd 500
 pt POINT <10,30>, <100,30>, <20,120>,<50,0>, <100,100> ;
 hBrush dd 0 ; комірка збереження дескриптора
  info db "Вы действительно хотите выйти из программы?",0
_Title db "Выход из программы",0
.data?
        hInstance dd ?
        CommandLine dd ?
        hWnd    dd ?
        left    dd ?
        top     dd ?
        hRgn dd ?
.code
start:
        invoke GetModuleHandle, NULL ; отримання дескриптора програми
        mov    hInstance,eax         ; збереження дескриптора програми
        invoke GetCommandLine
        mov CommandLine,eax
        invoke WinMain, hInstance,NULL,CommandLine, SW_SHOWDEFAULT
            invoke ExitProcess,eax

WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,\
```

```
                                                    CmdShow:DWORD
LOCAL wc    :WNDCLASSEX
LOCAL msg   :MSG
@<mov wc.cbSize, sizeof WNDCLASSEX>,<mov wc.lpfnWndProc, OFFSET
WndProc>
@<mov wc.style, CS_HREDRAW or CS_VREDRAW>,< mov wc.cbClsExtra, 0>
@<mov wc.cbWndExtra,0>,< mov eax, hInst>,< mov wc.hInstance, eax>
@<mov wc.hbrBackground,  COLOR_BTNFACE+1>,<mov
wc.lpszMenuName,   0>
@< mov wc.lpszClassName, offset szClassName>,<invoke LoadIcon, hInst, 0 >
mov wc.hIcon, eax    ; дескриптор піктограми
invoke LoadCursor, 0, IDC_ARROW ; курсор стандартний
@<mov wc.hCursor, eax>,<mov wc.hIconSm, 0>
  invoke RegisterClassEx, ADDR wc
  invoke GetSystemMetrics,SM_CXSCREEN ;  ширина екрана в пікселях
@<shr eax, 1>,<mov ebx, winWid>,<shr ebx, 1>,<sub eax, ebx>,<mov left, eax>
   invoke GetSystemMetrics,SM_CYSCREEN ; висота екрана в пікселях
@<shr eax, 1>,<mov ebx, winHgt>,<shr ebx, 1>,<sub eax, ebx>,<mov top, eax>
invoke CreateWindowEx,WS_EX_LEFT, ADDR szClassName, \
  ADDR szUntitled,  WS_OVERLAPPEDWINDOW,\
   left,top,winWid,winHgt, \ ; гориз. та верт. координати, X та Y вікна
    0,0, hInst,0
       mov   hWnd,eax
invoke ShowWindow,hWnd,SW_SHOWNORMAL ; стан показу вікна
invoke UpdateWindow,hWnd ; відновлення вікна через WM_PAINT
.WHILE TRUE
invoke GetMessage, ADDR msg,NULL,0,0 ; витягання повідомлення з
черги
         .BREAK .IF (!eax)
invoke DispatchMessage, ADDR msg ; відправка на обслуговування
    .ENDW
            mov     eax,msg.wParam
ret                              ; повернення з процедури WinMain
WinMain endp                     ; закінчення процедури з ім'ям WinMain

WndProc proc hWin   :DWORD, uMsg   :DWORD, wParam :DWORD, \
            lParam :DWORD
  LOCAL hdc:HDC
  LOCAL ps:PAINTSTRUCT
.if  uMsg == WM_CLOSE
invoke MessageBox,0,offset info,offset _Title,MB_YESNO
        .if eax == IDNO
          jmp  @wmpaind
        .endif
.ELSEIF  uMsg == WM_DESTROY ; обробка  повідомлення при знищенні
invoke PostQuitMessage, 0 ; відправлення повідомл. про знищення вікна
```

```
.ELSEIF uMsg==WM_PAINT ; якщо є повідомлення про перерисувування
@wmpaind:
 invoke BeginPaint,hWnd, ADDR ps    ; виклик підготовчої процедури та
                                                    ; заповнення структури
  mov  hdc,eax       ; збереження контексту

invoke CreatePolygonRgn,addr pt,5,WINDING ;
 mov hRgn,eax
invoke CreateSolidBrush,005B7FFFh ; створення щітки для фону
 mov hBrush,eax
invoke FillRgn,hdc,hRgn,hBrush ; заповнення області
                    ; без цієї функції з'являється повноцінне вікно
invoke SetWindowRgn, hWin,\ ; дескриптор вікна, яке буде змінюватися
     hRgn,\ ; дескриптор регіону
       1 ;  перерисовування
invoke EndPaint,hWnd, ADDR ps ; закінчення рисування
   .endif
invoke DefWindowProc,hWin, uMsg, wParam, lParam; відправка до
WndProc
      ret                                    ; повернення з процедури
WndProc endp                          ; закінчення процедури WndProc
end start                             ; закінчення програми з ім'ям start
```

Вигляд нестандартного вікна програми наведено на рисунку 2.2.

Рисунок 2.2 – Результат роботи програми

Координати точок в масиві наведені за таким порядком, при якому спочатку обирається одна точка лінії, а наступною парою координат указують закінчення цієї лінії.

Для знищення цього рисунка необхідно підвести курсор до верхньої частини рисунка, натиснути на праву клавішу мишки та вибрати "Закрыть" або натиснути ALT+F4. У результаті цієї дії відображається спрощене віконце з підтвердженням вибраної дії (рисунок 2.3).

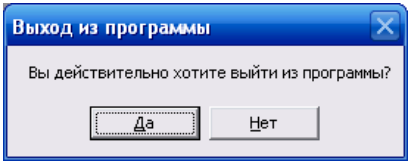


Рисунок 2.3 – Вигляд повідомлення з підтвердженням вибранної дії

Це віконце відображається в результаті обробці повідомлення WM_CLOSE фрагментом коду:

```
.if uMsg == WM_CLOSE
invoke MessageBox,0,offset info,offset _Title,MB_YESNO
        .if eax == IDNO         ; якщо вибрана кнопка “НЕТ”
          jmp  wmpaind
        .endif
```

У випадку, якщо не вибрана кнопка знищення фігури, то за міткою wmpaind здійснюється перехід на гілку обробки повідомлення WM_PAINT (для того щоб перерисувати фігуру).

У лістингу 2.3 наведено приклад програми, в якій створено 6-кутову фігуру, для управління якою використаються кнопки: кнопка x для знищення вікна, а кнопки x1 та x2 – для управління прозорістю вікна.

Лістинг 2.3:

```
.686                    ; директива визначення типу мікропроцесора
.model flat,stdcall  ; завдання лінійної моделі пам’яті та угоди ОС Windows
option casemap:none         ; відмінність малих та великих літер
;include \masm32\include\string.inc
include \masm32\include\windows.inc ; файли структур, констант …
include \masm32\macros\macros.asm
uselib user32,kernel32,gdi32, shell32, comctl32
@ MACRO a0,a1,a2,a3
a0
 a1
  a2
   a3
ENDM
IDI_ICON equ 22
  ButtonID equ 2  ; ідентифікатор кнопки "X"
  Button2ID equ 7 ;
   Button3ID equ 14 ;
```

```
.data
      _szT1 db "Прозрачность",0
      hwndButton HWND ?
  ButtonClassName db "button",0
  ButtonText db "X",0
  ButtonText1 db "X1",0
  ButtonText2 db "X2",0
    winx dd 200  ; Напівширина вікна
    winy dd 200  ; Напіввисота вікна
    sens dd 230  ; Початковий параметр прозорості
      ClassName db "SimpleWinClass",0
      AppName db "ZVEZDA",0
      MenuName db "FirstMenu",0
spt   POINT   <0,110>,<100,110>,<200,50>,<300,110>,<400,110>,<350,230>,
<400,370>, <300,370>, <200,450>,<100,370>, <0,370>,<50,230>
      spt1 RECT <54,79,341,296>       ; координати рамки 1
      spt2 RECT <52,77,343,298>       ; координати рамки 2
      wc WNDCLASSEX <>                ; ініціалізація
      msg MSG <>
      hwnd HWND ?
      hRgn dd ?
      hInstance HINSTANCE ?
.code
start:
  invoke GetModuleHandle, NULL ; отримання дескриптора програми
@<mov hInstance,eax>,<mov wc.cbSize,SIZEOF WNDCLASSEX>
@<mov wc.style,CS_HREDRAW or CS_VREDRAW>,<mov wc.cbClsExtra,0>
@<mov wc.lpfnWndProc, OFFSET WndProc>,< mov wc.cbWndExtra,0>
@<push hInstance >,< pop wc.hInstance >
invoke GetStockObject,WHITE_BRUSH ; Створення кольору
invoke CreateSolidBrush,0000FFE1h                ;  заливки
@<mov wc.hbrBackground,eax>,<mov wc.lpszMenuName,OFFSET MenuName>
  mov wc.lpszClassName,OFFSET ClassName ; ім'я класу

  invoke LoadIcon,NULL,IDI_ERROR                 ; іконка
@< mov wc.hIcon,eax >,< mov wc.hIconSm,eax>
invoke LoadCursor,NULL,IDC_CROSS                 ; курсор – хрест
  mov wc.hCursor,eax
  invoke RegisterClassEx,addr wc
invoke GetSystemMetrics,SM_CXSCREEN ; отримання ширини
   shr eax,1            ; знаходження центра
   sub eax,winx         ; отримання x-координати вікна
   mov winx,eax         ; збереження
invoke GetSystemMetrics,SM_CYSCREEN ; отримання висоти
   shr eax,1            ; знаходження центра
   sub eax,winy         ; отримання y-координати вікна
```

```
   sub eax,45         ; корекція на полігон
   mov winy,eax       ; збереження
invoke CreateWindowEx,0,ADDR ClassName, ADDR AppName, \
WS_OVERLAPPED,winx,winy, 600,600,0,0,hInstance,0 ;
  mov hwnd,eax                          ; збереження дескриптора
  invoke ShowWindow,hwnd,SW_SHOWNORMAL ; показати вікно
  invoke UpdateWindow,hwnd              ; відновити вікно
       .WHILE TRUE ; обробка повідомлень від окна
         invoke GetMessage,ADDR msg,0,0,0 ; читання повідомлення
           or eax,eax
           jz Quit
           invoke DispatchMessage,ADDR msg ; відправка на обслуговування
       .ENDW
       invoke DestroyWindow,hwnd
       Quit:
       mov eax,msg.wParam       ;очищення ресурсів
       invoke ExitProcess,eax    ;

WndProc proc hWnd:HWND,uMsg:UINT,wParam:WPARAM,\
                    lParam:LPARAM
       LOCAL hdc:HDC ;контекст
       LOCAL ps:PAINTSTRUCT ; структура для рисування
       LOCAL rect:RECT      ; структура координат
       mov eax,wParam       ; завантаження wParam
.IF uMsg==WM_PAINT      ; якщо рисування
         invoke GetDC,hWnd  ; отримати контекст за дескриптором
         mov hdc,eax        ; зберегти
invoke DrawEdge,hdc,ADDR spt1,EDGE_SUNKEN,BF_RECT ; рамка spt1
invoke DrawEdge,hdc,ADDR spt2,EDGE_SUNKEN,BF_RECT ; рамка spt2
       invoke BeginPaint,hWnd, ADDR ps ; початок рисування
       mov    hdc,eax
   invoke SetBkMode,hdc,0h            ; встановити фон тексту
   invoke SetTextColor,hdc,0ff0000h  ; встановлення кольору тексту
    invoke TextOut,hdc,153,30,addr _szT1,12 ; виведення тексту
       invoke EndPaint,hWnd, ADDR ps   ; завершення рисування
.ELSEIF uMsg==WM_DESTROY    ; якщо знищення
         invoke DeleteObject, hRgn     ;  видалити регіон
         invoke DestroyWindow,hWnd   ; видалення вікна
.ELSEIF uMsg==WM_CREATE      ; якщо створення
  m7:  invoke GetWindowLong,hWnd,GWL_EXSTYLE ; отримати стиль
       or eax,WS_EX_LAYERED      ; завантажити прозорість
 invoke SetWindowLong,hWnd,GWL_EXSTYLE,eax ; зміна атрибутів вікна
       invoke SetLayeredWindowAttributes, hWnd, 00FFFFFFh,\
           sens,LWA_ALPHA    ; застосувати прозорість до вікна
 invoke CreatePolygonRgn,addr spt,12,WINDING  ; створення
       invoke SetWindowRgn,hWnd,eax,1         ; та застосування полігона
```

```
  ;;Створення  дочірніх кнопок вікна hWnd
invoke CreateWindowEx,0, ADDR ButtonClassName,ADDR ButtonText,\
        WS_CHILD or WS_VISIBLE or BS_DEFPUSHBUTTON,\
            10,68,15,15,hWnd,ButtonID,hInstance,NULL ; кнопка "X"
invoke CreateWindowEx,0, ADDR ButtonClassName,ADDR ButtonText1,\
       WS_CHILD or WS_VISIBLE or BS_DEFPUSHBUTTON,\
            160,45,40,25,hWnd,Button2ID,hInstance,NULL ; кнопка "X1"
invoke CreateWindowEx,0, ADDR ButtonClassName,ADDR ButtonText2,\
       WS_CHILD or WS_VISIBLE or BS_DEFPUSHBUTTON,\
             200,45,40,25,hWnd,Button3ID,hInstance,NULL ; кнопка "X2"
       mov  hwndButton,eax
.ELSEIF uMsg==WM_COMMAND    ; якщо є повідомлення від меню
       mov eax,wParam
       .IF ax==ButtonID ; якщо повідомлення від кнопки закриття вікна "X"
        invoke PostQuitMessage,NULL
       .ELSEIF ax==Button2ID ; від кнопки прозорості "X1"
              mov sens,255 ; встановлення прозорості в одиницях
               jmp m7
       .ELSEIF ax==Button3ID ; від кнопки прозорості "X2"
             mov sens,190 ; встановлення прозорості в одиницях
               jmp m7
       .ENDIF                    ; закінчення логічної структури
.ELSEIF uMsg==WM_LBUTTONDOWN ; ліва  кнопка миші
       invoke SendMessage,hWnd,WM_NCLBUTTONDOWN, HTCAPTION,0
.ELSE
```

Вигляд нестандартного вікна програми наведено на рисунку 2.4.

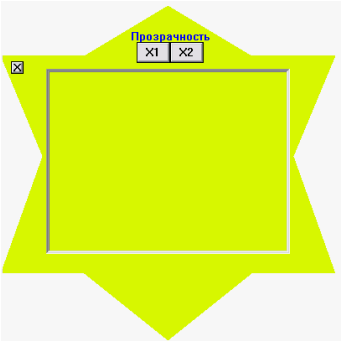


Рисунок 2.4 – Вигляд нестандартного вікна програми

Переміщення вікна (http://abyss-group.narod.ru/docs/articles/mousemove.htm) виконується в фрагменті коду:

```
.ELSEIF uMsg==WM_LBUTTONDOWN ; якщо натиснута ліва  кнопка миші
   invoke SendMessage,hWnd,WM_NCLBUTTONDOWN, HTCAPTION,0
```

У функції SendMessage повідомлення посилається вікну, і параметрами його є координати курсора миші. Повідомлення обробляється і повертається одне із значень, але для переміщення вікна використовуються тільки два: HTCLIENT та HTCAPTION. Після обробки повідомлення за умовчанням повертається результат області вікна, над якою була натиснута клавіша миші. Повідомлення HTCLIENT означає, що клавіша була натиснута над клієнтською областю, а повідомлення HTCAPTION – над заголовком вікна. Як відомо, Windows забезпечує перетягування за заголовок вікна, а отже, виконується перехоплення повідомлення по дорозі назад, і якщо клавіша була натиснута над клієнтською областю, то виконується підміна значення, що повертається, вказуючи що начеб-то натиснення було над заголовком.

При виконанні фрагмента коду:

```
…
hRgn1 dd ?
hRgn2 dd ?
…
.ELSEIF uMsg==WM_CREATE      ; якщо є повідомлення про створення
invoke CreateEllipticRgn,-150,-300,363,400 ; видима права велика частина еліпса
      mov hRgn1, eax;
invoke CreateEllipticRgn,363+150,-300,5,400; видима ліва велика частина еліпса
      mov hRgn2,eax;
invoke CombineRgn,hRgn1,hRgn1,hRgn2,RGN_AND ; накладення еліпсів
     invoke  CreateEllipticRgn,-40,-15,180,23 ; верхня ліва частина еліпса
      mov hRgn2,eax;
invoke  CombineRgn,hRgn1,hRgn1,hRgn2,RGN_DIFF  ;  віднімаємо верхню ліву частину
     ;; відрізання зверху праворуч
 invoke CreateEllipticRgn,180,-15,363+40,23 ; верхня права частина еліпса
      mov hRgn2,eax ;
      invoke  CombineRgn,hRgn1,hRgn1,hRgn2,RGN_DIFF;
invoke SetWindowRgn,hWnd,hRgn1,1; призначаємо отриманий регіон формі

;;Створення  дочірніх кнопок вікна hWnd
invoke CreateWindowEx,0, ADDR ButtonClassName,ADDR ButtonText1,\
     WS_CHILD or WS_VISIBLE or BS_DEFPUSHBUTTON,\
       153,280,50,23,hWnd,Button1ID,hInstance,0 ; кн. "Delete"
invoke CreateWindowEx,0, ADDR ButtonClassName,ADDR ButtonText4,\
```

```
    WS_CHILD or WS_VISIBLE or BS_DEFPUSHBUTTON,\
      135,240,84,23,hWnd,Button4ID,hInstance,0 ; "Расчеты"
          mov hwndButton,eax
.ELSEIF uMsg==WM_COMMAND    ; якщо є повідомлення від меню
    mov eax,wParam
    …
```

Отримаємо нестандартну форму у вигляді ламаних ліній (рисунок 2.5).

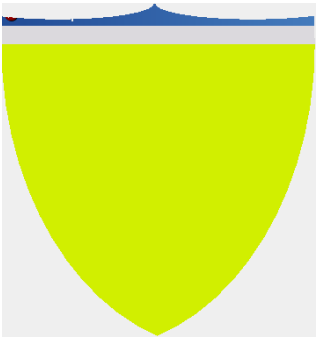


Рисунок 2.5 – Вигляд нестандартного вікна програми

2.2. Нестандартні вікна, подані у вигляді растрової картинки

У графічному файлі формату BMP інформація про картинку зберігається в растровому вигляді, тобто кожен піксель описаний певним кольором. Створення вікна нестандартної форми на основі растрової картинки полягає в накладенні зображення на діалогове вікно і видаленні всіх його регіонів, в яких знаходяться точки певного кольору. Цей колір вважається "прозорим", тому що справжню прозорість звичайний формат BMP не підтримує. Прозорим кольором фону вважається колір крапки в лівому верхньому кутку картинки. Непотрібні регіони необхідно видалити.

Картинку треба зберегти у форматі BMP, з глибиною кольору 8 бітів, оскільки в цьому випадку кількість кольорів у палітрі не перевищує 256, а кожна крапка описується одним байтом. Оскільки картинка накладатиметься на діалогове вікно, то зберігати її треба в ресурсах.

Для отримання кольору крапки використовується функція GetPixelColor (встановлює колір пікселя). У цій функції враховані особливості формату BMP, а саме те, що пікселі зберігаються відрядкові (від низу до верху), а кожен рядок зображення доповнюється нулями до довжини, кратної чотирьом байтам.

3. ПРИКЛАДИ СТВОРЕННЯ ПРОГРАМИ

3.1. Спіраль Архімеда з плавною зміною кольору фігури

Архімедова спіраль (http://ru.wikipedia.org/wiki/ Архимедова_ спираль) – спіраль, плоска крива, траєкторія крапки M (рисунок 3.1) яка рівномірно рухається уподовж лінії OV з початком у крапці O, тоді як сама лінія OV рівномірно обертається навколо крапки O.

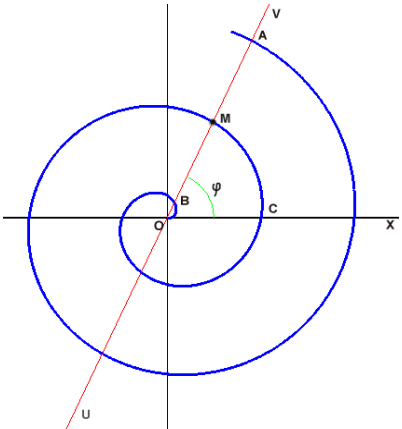


Рисунок 3.1 – Класична спіраль Архімеда

Іншими словами, відстань ρ = OM пропорційна куту повороту φ лінії OV. Повороту лінії OV на один і той же кут відповідає один і той же приріст ρ:

$$\rho = k\varphi,$$

де k – зсув крапки M по лінії r, при повороті на кут, який дорівнює одному радіану.

Повороту прямої на 2π відповідає зсув $a = |BM| = |MA| = 2k\pi$. Число a — називається кроком спіралі. Рівняння Архімедової спіралі у полярній системі координат можна переписати так:

$$P = (a/2\pi)\varphi.$$

Полярна система координат – двовимірна система координат, в якій кожна крапка на площині визначається двома числами – полярним кутом і полярним радіусом.

При обертанні лінії проти годинникової стрілки виходить права спіраль, при обертанні за годинниковою стрілкою – ліва спіраль. Позитивним значенням відповідає права спіраль, негативним – ліва

спіраль. Якщо точка М рухатиметься по прямій UV з негативних значень через центр обертання О і далі в позитивні значення уподовж прямій UV, то точка М опише обидві спіралі.

Відстані між крапками В та М, М та А дорівнюють кроку спіралі 2kπ. При розкручуванні спіралі відстань від точки О до точки М прагне до нескінченності. При цьому крок спіралі залишається постійним (кінцевим), тобто чим далі від центра, тим ближче витки спіралі за формою наближаються до кола.

Простіше використовувати (http://hijos.ru/2011/03/09/arximedova-spiral/) параметризацію спіралі (декартова система координат) за формулами:

$$x = \varphi \cos \varphi;$$
$$y = \varphi \sin \varphi$$

У декартовій або прямокутній системі координат відносини між координатами можна встановити тільки шляхом застосування тригонометричних рівнянь.

У лістингу 3.1 наведено файл ресурсів програми.

Лістинг 3.1. Файл ресурсів програми:

```
#define IDM_INFO 1
#define IDM_USL 3
#define IDM_EXIT 2
#define IDM_M1 11
#define IDM_M2 12
#define IDM_M3 13
#define IDM_V1 21
#define IDM_V2 22
#define IDM_V3 23
#define IDM_V4 24
#define IDM_V5 25
#define IDM_V6 26
#define IDM_V7 27
#define IDM_V8 28
#define IDM_V9 29
#define IDM_V0 20
#define IDI_ICON 1001
IDI_ICON ICON DISCARDABLE MOVEABLE LOADONCALL "Spiral1.ico"
FirstMenu MENU {
    POPUP "Программа"{
            MENUITEM "Условие...", IDM_USL
            MENUITEM "Об авторе...", IDM_INFO
            MENUITEM SEPARATOR
            MENUITEM "Выход", IDM_EXIT
    }
```

```
        POPUP "Масштаб"{
                MENUITEM "1:2", IDM_M1
                MENUITEM "1:1", IDM_M2
                MENUITEM "2:1", IDM_M3
        }
        POPUP "Количество витков"{
                MENUITEM "1", IDM_V1
                MENUITEM "2", IDM_V2
                MENUITEM "3", IDM_V3
                MENUITEM "4", IDM_V4
                MENUITEM "5", IDM_V5
                MENUITEM "6", IDM_V6
                MENUITEM "7", IDM_V7
                MENUITEM "8", IDM_V8
                MENUITEM "9", IDM_V9
                MENUITEM "10", IDM_V0
        }
    }
```

Програма виконання прикладу 3.1 наведена в лістингу 3.2.

Лістинг 3.2. Програма рисування спіралі Архімеда з плавною зміною кольору фігури:

```
    .686                    ; директива визначення типу мікропроцесора
.model flat,stdcall  ; завдання лінійної моделі пам'яті та угоди ОС Windows
    option casemap:none             ; відмінність малих та великих літер
WinMain proto :DWORD,:DWORD,:DWORD,:DWORD ; прототип процедури
     include \masm32\include\windows.inc   ; файли структур, констант …
     include \masm32\macros\macros.asm
     uselib user32, kernel32, fpu, gdi32
   @ MACRO b0,b1,b2,b3
   b0
    b1
     b2
      b3
   ENDM
.data                                   ; директива визначення даних
        ClassName db "Class", 0
        AppName db "Спираль Архимеда",0
        MenuName db "FirstMenu",0
        info_string db "Автор:  ",0
        usl_string db "Построение спирали Архимеда",0dh,0ah,
                "с количеством витков от 1 до 10",0dh,0ah,
                        "в трех вариантах масштаба",0
        info_caption db "Об авторе",0
        usl_caption db "Условие",0
```

```
        mas dd 07AB7h           ; кількість циклів
        two dd 2
        alpha dd 0.0            ; кутова координата
        delta dd 0.001          ; збільшення координати
        xdiv2 dd ?              ; середина по X та Y
        ydiv2 dd ?
        tmp dd 0                ; тимчасова змінна
        K1 dd 2.5               ; масштабні коефіцієнти
        K2 dd 5.0
        K3 dd 10.0
        divK dd 5.0
        xr dd 0                 ; координати функції
        yr dd 0
colour dd 32000   ; початкове значення кольору (зелений)
count db 0
.data?                          ; директива невизначених даних
    hInstance HINSTANCE ?
    CommandLine LPSTR ?
.const                          ; визначення констант
    IDM_INFO equ 1
    IDM_USL equ 3
    IDM_EXIT equ 2
    IDM_M1 equ 11               ; кількість виток спіралі
    IDM_M2 equ 12
    IDM_M3 equ 13
    IDM_V1 equ 21
    IDM_V2 equ 22
    IDM_V3 equ 23
    IDM_V4 equ 24
    IDM_V5 equ 25
    IDM_V6 equ 26
    IDM_V7 equ 27
    IDM_V8 equ 28
    IDM_V9 equ 29
    IDM_V0 equ 20
    IDI_ICON equ 1001
.code                           ; директива початку сегмента команд
start:                          ; мітка початку програми з ім'ям start
    invoke GetModuleHandle,NULL ; отримання дескриптора програми
    mov    hInstance,eax        ; збереження дескриптора програми
    invoke GetCommandLine
    mov CommandLine,eax
invoke WinMain,hInstance,NULL,CommandLine,SW_SHOWDEFAULT
invoke ExitProcess,eax ; повернення керування ОС Windows та
                                                ; визволення ресурсів
WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,\
```

```
  CmdShow:DWORD
 LOCAL wc:WNDCLASSEX ; резервування стека під структуру
 LOCAL msg:MSG      ; резервування стека під структуру MSG
 LOCAL hwnd:HWND ; резервування стека під хендл програми
 @<mov wc.cbSize,SIZEOF WNDCLASSEX>,<mov wc.cbClsExtra,0>
 @<mov wc.style,CS_HREDRAW or CS_VREDRAW >,<mov
 wc.cbWndExtra,0>
 @<mov wc.lpfnWndProc,OFFSET WndProc>,<push hInst>,<pop
 wc.hInstance >
@<mov wc.hbrBackground,COLOR_WINDOW+1>,<mov wc.lpszMenuName,OFFSET
MenuName>
       mov   wc.lpszClassName,OFFSET ClassName   ; ім'я класу
       invoke LoadIcon,hInstance,IDI_ICON              ; ресурс піктограми
 @< mov   wc.hIcon,eax >,< mov   wc.hIconSm,eax >
       invoke LoadCursor,NULL,IDC_ARROW    ; ресурс
       mov   wc.hCursor,eax            ; дескриптор курсора
 invoke RegisterClassEx,addr wc     ; реєстрація класу вікна
 INVOKE CreateWindowEx,\            ; функція створення вікна за зразком
       NULL,ADDR ClassName,\        ; стиль та адреса імені класу
       ADDR AppName,\               ; адреса імені вікна
       WS_OVERLAPPEDWINDOW,\              ; базовий стиль
       200,200,600,600\             ; x0, Y0, Δx, Δy координати вікна
       0,0,hInst,0  ; дескриптори батьківського вікна, меню, програми
        mov   hwnd,eax
 INVOKE ShowWindow, hwnd,SW_SHOWNORMAL ; видимість вікна
 INVOKE UpdateWindow, hwnd
       .WHILE TRUE                          ; поки істинне, то
 invoke GetMessage, ADDR msg, NULL, 0, 0 ; читання повідомлення
            or eax, eax                  ; формування ознак
            jz Quit                      ; перейти на мітку Quit, якщо eax = 0
     invoke DispatchMessage, ADDR msg ; відправка на обслуговування
       .ENDW                  ; закінчення циклу оброблення повідомлень
 Quit:
            mov   eax,msg.wParam
            ret                  ; повернення з процедури WinMain
     WinMain endp                        ; закінчення процедури з ім'ям WinMain

 WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM,\
  lParam:LPARAM
 LOCAL rect:RECT           ; резервування стека під структуру RECT
 LOCAL ps:PAINTSTRUCT ; резервування стека під структуру
 LOCAL hdc:HDC             ; резервування стека під хендл вікна
 .IF uMsg==WM_DESTROY ; якщо є повідомлення про знищення вікна
            invoke PostQuitMessage,NULL ; передача повідомлення WM_Quit
 .ELSEIF uMsg==WM_COMMAND              ; якщо є повідомлення з меню
            mov eax,wParam  ; формування ознак повідомлення з меню
```

```
        .IF ax==IDM_INFO          ; виведення інформації про автора
invoke MessageBox,NULL,ADDR info_string,ADDR info_caption,MB_OK
        .ELSEIF ax==IDM_USL               ; виведення умови
invoke MessageBox,NULL,ADDR usl_string,ADDR usl_caption,MB_OK
        .ELSEIF ax==IDM_M1 ; визначення коефіцієнта масштабу
@< mov ebx, K1 ; divK=2.5>,< mov divK, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_M2
@< mov ebx, K2 ; divK=5.0>,< mov divK, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_M3
        mov ebx, K3  ; divK=10.0
        mov divK, ebx
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V1 ; визначення кількості циклів
@< mov ebx, 0188Bh ; mas=1*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V2
@< mov ebx, 03116h ; mas=2*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V3
@< mov ebx, 049A1h ; mas=3*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V4
@< mov ebx, 0622Ch ; mas=4*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V5
@< mov ebx, 07AB7h ; mas=5*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V6
@< mov ebx, 09343h ; mas=6*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V7
@< mov ebx, 0ABCEh ; mas=7*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V8
@< mov ebx, 0C459h ; mas=8*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V9
@< mov ebx, 0DCE4h ; mas=9*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSEIF ax==IDM_V0
@< mov ebx, 0F56Fh ; mas=10*Pi*2000>,< mov mas, ebx >
        invoke InvalidateRect, hWnd, NULL, TRUE
        .ELSE
        invoke DestroyWindow,hWnd         ; знищення вікна
```

```
        .ENDIF
.ELSEIF uMsg==WM_PAINT ; якщо є повідомлення про перерисовування
invoke BeginPaint, hWnd, addr ps ; виклик підготовчої процедури
                                                ; та заповнення структури
      mov  hdc, eax                ; збереження контексту
invoke GetClientRect, hWnd, addr rect; занесення в структуру rect
                                                ; характеристик вікна
        mov alpha, 0               ; скидання кутової координати
        finit                      ; ініціювання співпроцесора
        fild rect.right            ; st(0) = (ширина вікна)
        fild two                   ; st(0) = 2
        fdivp st(1), st            ; st(0) = (ширина вікна)/2
        fistp dword ptr xdiv2      ; збереження середини по X
        fild rect.bottom           ; st(0) = (висота вікна)
        fild two                   ; st(0) = 2
        fdivp st(1), st            ; st(0) = (висота вікна)/2
        fistp dword ptr ydiv2      ; збереження середини по Y
invoke MoveToEx, hdc, xdiv2, ydiv2, NULL ; переміщення точки початку
                                          ; рисування у середину вікна
        mov ecx, mas     ; визначення кількості циклів
        push ecx         ; збереження в стеку кількості циклів
l1:     finit            ; ініціювання співпроцесора
        fld alpha        ; st(0) = alpha
        fcos             ; st(0) = cos(alpha)
        fld alpha        ; st(0) = alpha
        fmul divK        ; st(0) = alpha * divK, st(1) = cos(alpha)
        fmul             ; st(0) = st(0) * st(1) = alpha * divK * cos(alpha)
        frndint          ; округлення st(0)
        fild xdiv2
        fadd             ; додання середини по X
        fistp dword ptr xr  ; збереження X в зміну для виведення на екран
    fld alpha     ; st(0) = alpha
    fsin          ;st(0) = sin(alpha)
    fld alpha     ; st(0) = alpha
    fmul divK     ; st(0) = alpha * divK, st(1) = sin(alpha)
    fmul          ; st(0) = st(0) * st(1) = alpha * divK * cos(alpha)
    frndint       ; округлення st(0)
    fstp tmp
    fild ydiv2
    fsub tmp      ; віднімання від середини по Y
    fistp dword ptr yr       ; збереження X в зміну для виведення на екран
invoke SetPixel, hdc, xr, yr, colour ; рисування
     .IF  count==2           ; зміна кольору кожні X пікселя
        @<inc colour>,<mov count,0>
     .ELSE
        inc    count
```

```
        .ENDIF
            fld delta           ; збільшення кутової координати
            fld alpha
            fadd
            fstp alpha
            pop ecx             ; повернення з стека кількості циклів
            dec ecx             ; зменшення лічильника
            push ecx
            jz l2               ; продовження рисування
            jmp l1              ; вихід із циклу
l2:         pop ecx
            invoke EndPaint, hWnd, addr ps    ; закінчення рисування
            mov eax, 0
.ELSE                           ; інакше
invoke DefWindowProc,hWnd,uMsg,wParam,lParam   ; обробка та відправка
                                               ; повідомлення до функції WndProc
            ret
.ENDIF              ; закінчення логічної структури .IF - .ELSE
            xor eax,eax
            ret                         ; повернення з процедури
        WndProc endp                    ; закінчення процедури WndProc
        end start                       ; закінчення програми з ім'ям start
```

Лістинг 3.3. Командний bat-файл для створення виконавчого файлу:

```
ml /c /coff "spiral1.asm"
rc "spiral1.rc"
link /SUBSYSTEM:windows "spiral1.obj" "spiral1.res"
pause
start spiral1.exe
```

Результат виконання програми наведено на рисунку 3.2.

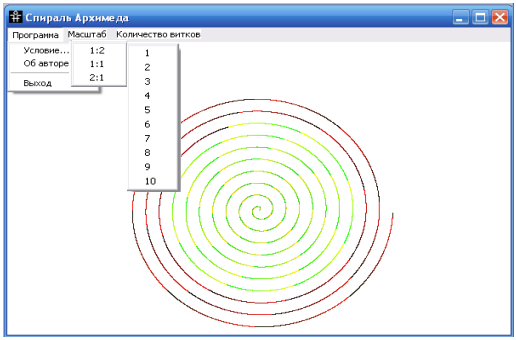


Рисунок 3.2 – Результат виконання програми з лістингу 3.1

Функція рисування точки SetPixel установлює колір точки з заданими координатами:

```
COLORREF WINAPI SetPixel(
  HDC hdc,          // контекст відображення
  int nXPos,        // x-координата точки
  int nYPos,        // y-координата точки
  COLORREF clrref); // колір точки
```

Кожні 2 пікселя змінюється колір виведених точок фігури. За їх зміну відповідає фрагмент коду:

```
.IF  count==2          ; зміна кольору кожні X пікселя
   @<inc colour>,<mov count,0>
.ELSE
   inc   count
.ENDIF
```

Початковим кольором фігури вибрано зелений колір зі значенням:

```
colour dd 32000   ; зелений – 7D00,
```

відносно якого здійснюються такі зміни.

3.2. Фігура у вигляді рози

Полярна роза (http://ru.wikipedia.org/wiki/Полярная_система_координат) – математична крива, схожа на квітку з пелюстками. Вона може бути визначена простим рівнянням у полярних координатах:

$$r(\varphi) = \alpha \cos(k\varphi + \theta_0)$$

для довільної постійної θ_0 (включаючи 0). Якщо k – ціле число, то це рівняння визначатиме розу з k пелюстками для непарних k, або з $2k$ пелюстками для парних k. Якщо k – раціональне, але не ціле, то графік, який заданий рівнянням, утворює фігуру подібну до троянди, але пелюстки перекриватимуться. Рози з 2, 6, 10, 14 і так далі пелюстками цим рівнянням визначити неможливо. Змінна б визначає довжину пелюсток.

Якщо вважати, що радіус не може бути негативним, то при будь-якому натуральному *до* виходить пелюсткова роза. Таким чином, рівняння $r(\varphi) = \alpha \cos 2\varphi$ визначатиме розу з двома пелюстками. З геометричної точки зору радіус – це відстань від полюса до крапки, і він не може бути негативним.

У програмі створюються два вікна: у першому вікні виконується рисування фігури рози, а в другому вікні – виводиться фотографія.

Лістинг 7.4. Файл ресурсів програми створення рози:

```
#define IDM_EXIT   1 // "Выход"
#define IDM_AUTOR  2 // "Об авторе"
```

```
#define IDM_HELP   3 // "О программе"
#define IDM_12 4   // Масштаб 1:2
#define IDM_11 5   // Масштаб 1:1
#define IDM_21 6   // Масштаб 2:1
#define IDM_COL1 7   //  "Синий"
#define IDM_COL2 8   //  "Зеленый"
#define IDM_COL3 9   //  "Красный"
#define IDM_COL4 10  //  "Черный"
#define IDM_FOTO 11 //
#define IDM_K1 13   // Кількість пелюстків  – 1
#define IDM_K2 14   // Кількість пелюстків  – 2
#define IDM_K3 15  //  Кількість пелюстків  – 3
#define IDM_K4 16  //  Кількість пелюстків  – 4
#define IDM_K5 17  //  Кількість пелюстків  – 5
#define IDM_K6 18  //  Кількість пелюстків  – 6
#define IDM_K7 19  //  Кількість пелюстків  – 7
#define IDI_ICON 22 //
#define IDB_PIC 23 //
IDB_PIC BITMAP "foto.bmp"
IDI_ICON ICON DISCARDABLE MOVEABLE LOADONCALL  "Roza.ico"
 FirstMenu MENU {
    POPUP "Программа"{
      MENUITEM "О программе",IDM_HELP
     MENUITEM SEPARATOR
     MENUITEM "Выход",IDM_EXIT
           }
 POPUP "Параметры"{
 POPUP "Масштаб"{
        MENUITEM "1:2", IDM_12
        MENUITEM "1:1", IDM_11
        MENUITEM "2:1", IDM_21
              }
 POPUP "Цвет"{
        MENUITEM "Красный", IDM_COL1
        MENUITEM "Зеленый", IDM_COL2
        MENUITEM "Синий", IDM_COL3
        MENUITEM "Черный", IDM_COL4
              }
 POPUP "Кол-во лепестков"{
        MENUITEM "1", IDM_K1
        MENUITEM "2", IDM_K2
        MENUITEM "3", IDM_K3
        MENUITEM "4", IDM_K4
        MENUITEM "5", IDM_K5
        MENUITEM "6", IDM_K6
        MENUITEM "7", IDM_K7
```

```
                }
        }
         POPUP "Автор"{
           MENUITEM "Об авторе", IDM_AUTOR
           MENUITEM "Фото", IDM_FOTO
         }
        }
```

Лістинг 3.5. Файл програми створення рози:

```
      .686            ; директива визначення типу мікропроцесора
.model flat,stdcall ; завдання лінійної моделі пам'яті та угоди ОС Windows
option casemap:none        ; відмінність малих та великих літер
WinMain proto :DWORD,:DWORD,:DWORD,:DWORD ; прототип процедури
include \masm32\include\windows.inc          ; файли структур, констант …
include \masm32\macros\macros.asm
uselib masm32, user32, kernel32,gdi32, fpu,shell32
.data                                        ; директива визначення даних
            ClassName db "Class", 0
            ClassName2 db "Class2", 0
            AppName db "Роза",0
            ChildName db "Фото автора",0
            MenuName db "FirstMenu",0
            info_string db " КИТ ",0
usl_string db "Курсовой проект по системному
программированию.",0dh,0ah,
            "Тема: Построение по геометрической формуле розы c",0dh,0ah,
            "изменением цвета, в трех вариантах масштаба.",0
            info_caption db "Автор программы",0
            usl_caption db "О программе",0
red RGBQUAD <255,0,0>
green RGBQUAD <0,255,0>
blue RGBQUAD <0,0,255>
black RGBQUAD <0,0,0>
  col dd 0
  arg dt 0.01        ; аргумент збільшення вугла
  pi dt ?   ; константа
  fi dt 0    ; вугол фі
  ci dt 200.0
  k dt 5.0
k1 dt 1.0
k2 dt 1.5
k3 dt 2.0
k4 dt 2.5
k5 dt 3.0
k6 dt 3.5
k7 dt 4.0
```

```
  tw dt 2.0          ; два
  temp dt ?          ; тимчасова змінна
  x dt ?   ; x для розрахунків
  y dt ?   ; y для розрахунків
  xr dd 0 ; x для рисування
  yr dd 0  ; y для рисування
  cicl dd ? ; змінна збереження потокової кількості циклів   two dd 2 ; два
  xt dd 0  ; x для переведення
  yt dd 0 ; y для переведення
          xdiv2 dd ?          ; середина по X
          ydiv2 dd ?           ; середина по Y
          ci1 dt 100.0        ; масштабний коефіцієнт
          ci2 dt 200.0    ; масштабний коефіцієнт
          ci3 dt 300.0    ; масштабний коефіцієнт
   hchild dd ?                ; дескриптор дочірнього вікна
   hBitmap dd ?   ; дескриптор зображення
   hwnd dd ?      ; дескриптор батьківського вікна
   note NOTIFYICONDATA <>  ; структура іконки на панелі tray
   color RGBQUAD <>
.data?                                  ; директива невизначених даних
    hInstance HINSTANCE ?
    CommandLine LPSTR ?
    .const                                      ; визначення констант
    IDM_EXIT  equ 1 ; "Выход"
    IDM_AUTOR equ 2 ; "Об авторе"
    IDM_HELP equ  3 ; "О программе"
    IDM_12 equ 4 ; Масштаб 1:2
    IDM_11 equ 5 ; Масштаб 1:1
    IDM_21 equ 6 ; Масштаб 2:1
    IDM_COL1 equ 7  ; "Синий"
    IDM_COL2 equ 8  ; "Зеленый"
    IDM_COL3 equ 9  ; "Красный"
    IDM_COL4 equ 10 ; "Черный"
IDM_FOTO equ 11 ;
IDM_K1 equ 13 ; Кількість пелюстків – 1
IDM_K2 equ 14 ; Кількість пелюстків – 2
IDM_K3 equ 15 ; Кількість пелюстків – 3
IDM_K4 equ 16 ; Кількість пелюстків – 4
IDM_K5 equ 17 ; Кількість пелюстків – 5
IDM_K6 equ 18 ; Кількість пелюстків – 6
IDM_K7 equ 19 ; Кількість пелюстків – 7
    IDI_ICON equ 22
    IDB_PIC equ 23
    IDI_TRAY equ 0
    WM_SHELLNOTIFY equ 5
.code                           ; директива початку сегмента команд
```

```
start:                          ; мітка початку програми з ім'ям start
invoke GetModuleHandle,NULL  ; отримання дескриптора програми
        mov    hInstance,eax        ; збереження дескриптора програми
        invoke GetCommandLine
        mov CommandLine,eax
invoke WinMain,hInstance,NULL,CommandLine,SW_SHOWDEFAULT
invoke ExitProcess,eax ; повернення керування ОС та визволення
ресурсів

WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,\
                                CmdShow:DWORD
LOCAL wc:WNDCLASSEX ; резервування стека під структуру
LOCAL wc2:WNDCLASSEX ; резервування стека під структуру
LOCAL msg:MSG           ; резервування стека під структуру MSG
mov wc.cbSize,SIZEOF WNDCLASSEX ; кількість байтів структури
mov wc.style,CS_HREDRAW or CS_VREDRAW ; стиль та поведінка вікна
mov wc.lpfnWndProc,OFFSET WndProc ; адреса процедури WndProc
mov wc.cbClsExtra,NULL  ; кількість байтів для структури класу
mov wc.cbWndExtra,NULL ; кількість байтів для структури вікна
        push  hInst         ; збереження в стеку дескриптора програми
        pop   wc.hInstance ; повернення дескриптора в поле структури
        mov   wc.hbrBackground,  COLOR_WINDOW+1 ; колір вікна
        mov   wc.lpszMenuName,OFFSET MenuName   ; ім'я ресурсу меню
        mov   wc.lpszClassName,OFFSET ClassName   ; ім'я класу
        invoke LoadIcon,hInstance,IDI_ICON          ; ресурс піктограми
        mov   wc.hIcon,eax          ; дескриптор піктограми
        mov   wc.hIconSm,eax      ; дескриптор маленького віконця
        invoke LoadCursor,NULL,IDC_ARROW    ; ресурс
        mov   wc.hCursor,eax                ; дескриптор курсора
        invoke RegisterClassEx,addr wc      ; реєстрація класу вікна
INVOKE CreateWindowEx,\ ; функція створення вікна за зразком
        NULL,ADDR ClassName,\ ; стиль та адреса імені класу
        ADDR AppName,\          ; адреса імені вікна
        WS_OVERLAPPEDWINDOW,\  ; базовий стиль
        200,200,\ ; горизизонтальна та вертикальна координати вікна
        600,600,\           ; ширина та висота вікна
        0,0,hInst,0  ; дескриптори батьківського вікна, меню, програми
        mov   hwnd,eax
; для завантаження фото
mov   wc2.cbSize,SIZEOF WNDCLASSEX ; кількість байтів структури
mov   wc2.style,CS_BYTEALIGNWINDOW ; стиль та поведінка вікна
mov   wc2.lpfnWndProc,OFFSET WndProc2 ; адреса процедури WndProc
mov   wc2.cbClsExtra,NULL           ; кількість байтів для структури класу
mov   wc2.cbWndExtra,NULL           ; кількість байтів для структури вікна
push  hInst                 ; збереження в стеку дескриптора програми
pop   wc2.hInstance         ; повернення дескриптора в поле структури
```

```
mov   wc2.hbrBackground,COLOR_BTNFACE+1; колір вікна
          mov   wc2.lpszMenuName,NULL     ; ім'я ресурсу меню
          mov   wc2.lpszClassName,OFFSET ClassName2       ; ім'я класу
invoke LoadIcon,hInstance,IDI_ICON          ; ресурс піктограми
          mov   wc2.hIcon,eax         ; дескриптор піктограми
          mov   wc2.hIconSm,eax     ; дескриптор маленького віконця
          invoke LoadCursor,NULL,IDC_ARROW    ; ресурс
          mov   wc2.hCursor,eax       ; дескриптор курсора
          invoke RegisterClassEx,addr wc2    ; реєстрація класу вікна
INVOKE ShowWindow, hwnd,SW_SHOWNORMAL ; видимість вікна
INVOKE UpdateWindow, hwnd
          .WHILE TRUE                         ; поки істинне, то
   invoke GetMessage, ADDR msg, NULL, 0, 0 ; читання повідомлення
            or eax, eax                       ; формування ознак
            jz Quit                       ; перейти на мітку Quit, якщо eax = 0
 invoke DispatchMessage, ADDR msg ; відправка на обслуговування
          .ENDW              ; закінчення циклу оброблення повідомлень
Quit:
          mov   eax,msg.wParam
          ret                ; повернення з процедури WinMain
WinMain endp                 ; закінчення процедури з ім'ям WinMain

WndProc proc hWnd:HWND, uMsg:UINT,
wParam:WPARAM,lParam:LPARAM
LOCAL rect:RECT            ; резервування стека під структуру RECT
LOCAL ps:PAINTSTRUCT ; резервування стека під структуру
LOCAL hdc:HDC              ; резервування стека під хендл вікна
.IF uMsg==WM_DESTROY ; якщо є повідомлення про знищення вікна
          invoke PostQuitMessage,NULL ; передача повідомлення WM_Quit
.ELSEIF uMsg==WM_CREATE
  mov color.rgbRed,0
  mov color.rgbGreen,0
  mov color.rgbBlue,0
  mov note.cbSize,sizeof NOTIFYICONDATA ; розмір структури
  push hwnd
  pop note.hwnd                ; хендл вікна
  mov note.uID,IDI_TRAY      ; ідентифікатор іконки
  mov note.uFlags,NIF_ICON+NIF_MESSAGE+NIF_TIP ; всі поля структури
  mov note.uCallbackMessage,WM_SHELLNOTIFY ; користувацьке
                                                          повідомлення
invoke LoadIcon,hInstance,IDI_ICON ; завантаження хендла іконки
          mov note.hIcon,eax
invoke lstrcpy,ADDR note.szTip,ADDR AppName ; спливаюче підказування
invoke Shell_NotifyIcon,NIM_ADD,ADDR note ; додавання іконки в трей
.ELSEIF uMsg==WM_COMMAND             ; якщо є повідомлення з меню
          mov eax,wParam  ; формування ознак повідомлення з меню
```

```
.IF ax==IDM_AUTOR       ; виведення інформації про автора
  invoke MessageBox,NULL,ADDR info_string,ADDR info_caption,MB_OK
.ELSEIF ax==IDM_HELP              ; виведення умови
  invoke MessageBox,NULL,ADDR usl_string,ADDR usl_caption,MB_OK
.ELSEIF ax==IDM_12 ; Масштаб 1:2
invoke FpuSub, ADDR ci,ADDR ci,ADDR ci, SRC1_REAL or SRC2_REAL or
DEST_MEM
invoke FpuAdd, ADDR ci,ADDR ci1,ADDR ci, SRC1_REAL or SRC2_REAL
or DEST_MEM
          invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_11 ; Масштаб 1:1
invoke FpuSub, ADDR ci,ADDR ci,ADDR ci, SRC1_REAL or SRC2_REAL or
DEST_MEM
invoke FpuAdd, ADDR ci,ADDR ci2,ADDR ci, SRC1_REAL or SRC2_REAL
or DEST_MEM
          invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_21 ; Масштаб 2:1
invoke FpuSub, ADDR ci,ADDR ci,ADDR ci, SRC1_REAL or SRC2_REAL or
DEST_MEM
invoke FpuAdd, ADDR ci,ADDR ci3,ADDR ci, SRC1_REAL or SRC2_REAL
or DEST_MEM
          invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна

.ELSEIF ax==IDM_K1 ; кількість пелюстків  – 1
invoke FpuSub, ADDR k,ADDR k,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
invoke FpuAdd, ADDR k,ADDR k1,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
          invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_K2 ; кількість пелюстків  – 2
invoke FpuSub, ADDR k,ADDR k,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
 invoke FpuAdd, ADDR k,ADDR k2,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
          invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_K3 ; кількість пелюстків  – 3
invoke FpuSub, ADDR k,ADDR k,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
invoke FpuAdd, ADDR k,ADDR k3,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
          invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_K4 ; кількість пелюстків  – 4
invoke FpuSub, ADDR k,ADDR k,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
 invoke FpuAdd, ADDR k,ADDR k4,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
```

```
        invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_K5 ; кількість пелюстків – 5
invoke FpuSub, ADDR k,ADDR k,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
 invoke FpuAdd, ADDR k,ADDR k5,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
        invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_K6 ; кількість пелюстків – 6
invoke FpuSub, ADDR k,ADDR k,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
invoke FpuAdd, ADDR k,ADDR k6,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
        invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_K7 ; кількість пелюстків – 7
invoke FpuSub, ADDR k,ADDR k,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
invoke FpuAdd, ADDR k,ADDR k7,ADDR k, SRC1_REAL or SRC2_REAL or
DEST_MEM
        invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_COL1 ; встановлення кольору та  WM_PAINT
  mov color.rgbRed,0
  mov color.rgbGreen,0
  mov color.rgbBlue,255
        invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_COL2 ; встановлення кольору та  WM_PAINT
  mov color.rgbRed,0
  mov color.rgbGreen,255
  mov color.rgbBlue,0
        invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_COL3 ; встановлення кольору та  WM_PAINT
  mov color.rgbRed,255
  mov color.rgbGreen,0
  mov color.rgbBlue,0
        invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_COL4 ; встановлення кольору  WM_PAINT
  mov color.rgbRed,0
  mov color.rgbGreen,0
  mov color.rgbBlue,0
        invoke InvalidateRect, hWnd, NULL, TRUE ; перерисовування вікна
.ELSEIF ax==IDM_FOTO
INVOKE CreateWindowEx,\ ; функція створення вікна за зразком
 NULL,ADDR ClassName2,\          ; стиль та адреса імені класу
 ADDR ChildName,\                ; адреса імені вікна
  WS_OVERLAPPEDWINDOW,\    ; базовий стиль
  200,200,350,400,\          ;  координати вікна
 hwnd,0,hInstance,0  ; дескриптори батьківського вікна, меню, програми
```

```
            mov   hchild,eax
  INVOKE ShowWindow, hchild,SW_SHOWNORMAL ; видимість вікна
            INVOKE UpdateWindow, hchild
.ELSE
              invoke DestroyWindow,hWnd        ; знищення вікна
.ENDIF
.ELSEIF uMsg==WM_PAINT ; якщо є повідомлення про перерисовування
invoke BeginPaint, hWnd, addr ps ; підготовча проц. та заповнення структ.
  mov     hdc, eax              ; збереження контексту
invoke GetClientRect, hWnd, addr rect; занесення характ.  вікна
invoke FpuSub, ADDR fi,ADDR fi,ADDR fi, SRC1_REAL or SRC2_REAL or
DEST_MEM ; встановлення кута на 0
            finit                           ; ініціювання співпроцесора
            fild rect.right                 ; st(0) = (ширина вікна)
            fild two                        ; st(0) = 2
            fdivp st(1), st                 ; st(0) = (ширина вікна)/2
            fistp dword ptr xdiv2           ; збереження середини по X
            fild rect.bottom                ; st(0) = (висота вікна)
            fild two                        ; st(0) = 2
            fdivp st(1), st                 ; st(0) = (висота вікна)/2
            fistp dword ptr ydiv2           ; збереження середини по Y
            mov ecx, 36000   ; встановлення кількості циклів
            mov cicl,ecx        ; занесення кількості циклів у змінну
l1:
invoke FpuMul, ADDR k, ADDR fi, 0, SRC1_REAL or SRC2_REAL or
DEST_FPU
invoke FpuSin, 0,0, SRC1_FPU or DEST_FPU
invoke FpuMul, 0, ADDR ci, ADDR pi, SRC1_FPU or SRC2_REAL or
DEST_MEM
invoke FpuCos, ADDR fi,0, SRC1_REAL or DEST_FPU ;st(0)=cos(fi)
invoke FpuMul, 0, ADDR pi, ADDR x, SRC1_FPU or SRC2_REAL or
DEST_MEM
                                                ; x=pi*cos(fi)
invoke FpuSin, ADDR fi,0, SRC1_REAL or DEST_FPU  ; st(0)=sin(fi)
invoke FpuMul, 0, ADDR pi, ADDR y, SRC1_FPU or SRC2_REAL or
DEST_MEM
                                                 ; x=pi*sin(fi)
 fld x
 frndint              ; округлення st(0)
 fistp dword ptr xt
 fld y
 frndint              ; округлення st(0)
 fistp dword ptr yt
; зміщення точки відповідно до центра вікна
  mov eax,xt
  add eax,xdiv2
```

```
  mov xr,eax
  mov eax,yt
  add eax,ydiv2
  mov yr,eax
invoke SetPixel, hdc, xr, yr, dword ptr color ; рисування точки
invoke FpuAdd, ADDR fi, ADDR arg, ADDR fi, SRC1_REAL or SRC2_REAL
or DEST_MEM ; кут
  mov ecx,cicl   ; занесення кількості циклів у регістр
          dec ecx          ; зменшення лічильника
          mov cicl,ecx  ; занесення кількості циклів у змінну
          push ecx   ; занесення кількості циклів у стек
          jz l2            ; вихід з циклу
          jmp l1           ; продовження рисування
l2:
          invoke EndPaint, hWnd, addr ps      ; закінчення рисування
          mov eax, 0
.ELSE                      ; інакше
invoke DefWindowProc,hWnd,uMsg,wParam,lParam
          ret
.ENDIF          ; закінчення логічної структури .IF - .ELSE
          xor eax,eax
          ret                        ; повернення з процедури
WndProc endp                         ; закінчення процедури WndProc
WndProc2 proc hWin:DWORD,uMsg:DWORD,wParam :DWORD,lParam
:DWORD
LOCAL rect:RECT          ; резервування стека під структуру RECT
LOCAL ps:PAINTSTRUCT ; резервування стека під структуру
LOCAL hdc:HDC            ; резервування стека під хендл вікна
LOCAL hMemDC:HDC      ; резервування стека під хендл вікна
.if uMsg == WM_CREATE
invoke LoadBitmap,hInstance,IDB_PIC ;отримання хендла зображення
  mov hBitmap,eax
.elseif uMsg == WM_PAINT
invoke BeginPaint,hchild,ADDR ps ; виклик процедури та заповнення ps
  mov hdc,eax
invoke CreateCompatibleDC,hdc ; створення контексту памяті
  mov hMemDC,eax
invoke SelectObject,hMemDC,hBitmap ; вибір у контекст
invoke GetClientRect,hchild,ADDR rect ; визначення розміру вікна
invoke BitBlt,hdc,0,0,rect.right,rect.bottom,hMemDC,0,0,SRCCOPY
                                                                    ; виведення
invoke DeleteDC,hMemDC ; видалення дескриптора
invoke EndPaint,hchild,ADDR ps ; закінчення рисування
.elseif uMsg == WM_DESTROY
invoke DeleteObject,hBitmap
invoke PostQuitMessage,NULL ; передача повідомлення WM_Quit
```

```
.endif
invoke DefWindowProc,hWin,uMsg,wParam,lParam
  ret
WndProc2 endp
end start                    ; закінчення програми з ім'ям start
```

Результат виконання програми наведено на рисунку 3.3.

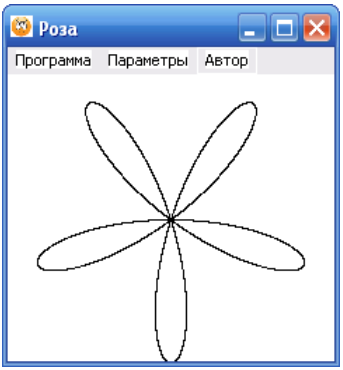


Рисунок 3.3 – Результат виконання програми з лістингу 3.1

3.3. Створення квадратів, що рухаються

У програмі створено два квадрати, які одночасно рухаються за годинниковою стрілкою. Файл ресурсів програми рисування квадратів наведено в лістингу 3.6.

Лістинг 3.6. Файл ресурсів програми рисування квадратів:

```
#define IDC_BUTTON 3001
#define IDM_SQUARES 32000
#define IDM_AVTOR 32001
#define IDM_FOTO 32002
#define IDM_TASK 32003
#define IDM_EXIT 32004
MyMenu MENU
 {
     POPUP "Задание"
     {
          MENUITEM "Квадраты",IDM_SQUARES
          MENUITEM "Задание",IDM_TASK
```

```
                MENUITEM SEPARATOR
                MENUITEM "Выход",IDM_EXIT
            }
            POPUP "О авторе"
            {
                MENUITEM "Информация ",IDM_AVTOR
                MENUITEM "Фотография",IDM_FOTO
            }
        }
        MyDialog DIALOGEX 0,0,300,245
        CAPTION "Курсовая работа"
        FONT 0,""
        MENU MyMenu
        CLASS "DLGCLASS"
        STYLE 0x0004 | DS_CENTER | WS_CAPTION | WS_VISIBLE |
;** створення діалогового вікна з модальною рамкою діалогового вікна **
        DS_MODALFRAME | DS_3DLOOK
        EXSTYLE 0x00000000
        BEGIN
CONTROL
"OK",IDC_BUTTON,"Button",0x10000001,250,230,46,13,0x00000000
        END
500 ICON MOVEABLE PURE LOADONCALL DISCARDABLE "icon.ico"
750 BITMAP MOVEABLE PURE LOADONCALL DISCARDABLE "foto.bmp"
```

Файл програми рисування квадратів наведено в лістингу 3.7.

Лістинг 3.7. Файл програми рисування квадратів:

```
        .686            ; директива визначення типу мікропроцесора
.model flat,stdcall ; завдання лінійної моделі пам'яті та угоди ОС Windows
option casemap:none         ; відмінність малих та великих літер
WinMain proto :DWORD,:DWORD,:DWORD,:DWORD ; прототип процедури
include \masm32\include\windows.inc          ; файли структур, констант …
include \masm32\macros\macros.asm
uselib masm32, user32, kernel32,gdi32, fpu,shell32
        ShowSquares  equ 0
        ShowAvtor    equ 1
        ShowFoto     equ 2
        ShowTask     equ 3
.data
        ClassName db "DLGCLASS", 0
        MenuName db "MyMenu", 0
        DlgName db "MyDialog", 0
        statClass db "STATIC", 0
        Mode dd ShowSquares
        Avtor1  db "Автор: Кальчева Екатерина, номер группы – КИТ 19а", 0
```

```
        Avtor2 db "e-mail shweps_665@mail.ru", 0
Task1  db "  Нарисовать изображение которое состоит из вложенных", 0
Task2  db "  один в другой квадрат, которые динамически вращаются", 0
Task3  db "Требования:", 0
Task4  db "  комментарии к коду программы", 0
Task5  db "  вывод в диалоговое окно с элементами меню под XP", 0
Task6  db "  в меню информация об авторе и фотография + задание", 0
Task7  db "  в верхней части созданного окна отобразить иконку", 0
Task8  db "  в правом нижнем отобразить иконку программы", 0
Task9  db "  текст и фон за текстом должны быть цветными", 0
.data?
        hInstance HINSTANCE ?
        wc WNDCLASSEX <?>
        msg MSG <?>
        hDlg HWND ?
        hIcon HWND ?
        hFotoImage HWND ?
.const
        IDC_BUTTON          equ 3001
        IDM_SQUARES         equ 32000
        IDM_AVTOR           equ 32001
        IDM_FOTO  equ 32002
        IDM_TASK   equ 32003
        IDM_EXIT    equ 32004
.code
;**** функція рисування квадратів ***************************************
        DrawSquares proc hWnd:DWORD, hDC:DWORD
            LOCAL time:DWORD
            LOCAL radius:DWORD
            LOCAL centerx, centery:DWORD
            LOCAL rx:DWORD
            LOCAL ry:DWORD
            LOCAL time_scale:DWORD
            LOCAL number:DWORD
            LOCAL rect:RECT
; рисування  квадратів
            .IF Mode == ShowSquares
                    ; визначення розміру вікна
            INVOKE GetClientRect, hWnd, ADDR rect
                    mov eax, rect.bottom
; визначення розміру малого квадрата
                    mov radius, eax
                    shr radius, 3
; визначення координат центра вікна
                    mov centery, eax
                    shr centery, 1
```

```
                mov eax, rect.right
                mov centerx, eax
                shr centerx, 1
; змінна циклу – номер квадрата від 0 до 1
                mov number, 0
; цикл для рисування двох квадратів
        again:
; визначення кута повороту квадрата залежно від потокового часу
        INVOKE GetTickCount
                mov time, eax
                finit
                fild time
                mov time_scale, 1000
                fidiv time_scale
    ; визначення зміщення для вершин квадрата
                fsincos
                fimul radius
                fistp rx
                fimul radius
                fistp ry
    ; переміщення потокової позиції у вершину 1
                mov eax, centerx
                add eax, rx
                mov ebx, centery
                add ebx, ry
                INVOKE MoveToEx, hDC, eax, ebx, 0
    ; рисування  лінії між вершинами 1 та 2
                mov eax, centerx
                sub eax, ry
                mov ebx, centery
                add ebx, rx
                INVOKE LineTo, hDC, eax, ebx
    ; рисування  лінії між вершинами 2 та 3
                mov eax, centerx
                sub eax, rx
                mov ebx, centery
                sub ebx, ry
                INVOKE LineTo, hDC, eax, ebx
    ; рисування лінії між вершинами 3 та  4
                mov eax, centerx
                add eax, ry
                mov ebx, centery
                sub ebx, rx
                INVOKE LineTo, hDC, eax, ebx
    ; рисування  лінії між вершинами 4 та 1
                mov eax, centerx
```

```
                add eax, rx
                mov ebx, centery
                add ebx, ry
                INVOKE LineTo, hDC, eax, ebx
; подвоювання розміру для наступного квадрата
                shl radius, 1
; перевірка змінної циклу для двох квадратів
                inc number
                cmp number, 2
                jl again
    .ENDIF
    ret
DrawSquares endp

;**** функція відображення інформації про автора ********************
DrawAvtor proc hWnd:DWORD, hDC:DWORD
    .IF Mode == ShowAvtor
                INVOKE SetTextColor, hDC, 00ffffh
                INVOKE SetBkColor, hDC, 808000h
                INVOKE StrLen, ADDR Avtor1
                INVOKE TextOut, hDC, 10, 60, ADDR Avtor1, eax
        INVOKE StrLen, ADDR Avtor2
        INVOKE TextOut, hDC, 10, 80, ADDR Avtor2,eax
    .ENDIF
    ret
DrawAvtor endp

;**** функція відображення завдання **********************************
DrawTask proc hWnd:DWORD, hDC:DWORD
    .IF Mode == ShowTask
INVOKE SetTextColor, hDC, 0FF0000h
INVOKE SetBkColor, hDC, 00000FFh
INVOKE StrLen, ADDR Task1
INVOKE TextOut, hDC, 10, 60, ADDR Task1, eax
INVOKE StrLen, ADDR Task2
INVOKE TextOut, hDC, 10, 80, ADDR Task2, eax
INVOKE StrLen, ADDR Task3
INVOKE TextOut, hDC, 10, 100, ADDR Task3, eax
INVOKE StrLen, ADDR Task4
INVOKE TextOut, hDC, 10, 120, ADDR Task4, eax
INVOKE StrLen, ADDR Task5
INVOKE TextOut, hDC, 10, 140, ADDR Task5, eax
INVOKE StrLen, ADDR Task6
INVOKE TextOut, hDC, 10, 160, ADDR Task6, eax
INVOKE StrLen, ADDR Task7
INVOKE TextOut, hDC, 10, 180, ADDR Task7, eax
```

```
        INVOKE StrLen, ADDR Task8
        INVOKE TextOut, hDC, 10, 200, ADDR Task8, eax
        INVOKE StrLen, ADDR Task9
        INVOKE TextOut, hDC, 10, 220, ADDR Task9, eax
            .ENDIF
            ret
        DrawTask endp

        ;**** функція обробки повідомлень ***********************************
WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM,\
                    lParam:LPARAM
            LOCAL hDC:DWORD
            LOCAL rct:RECT
            LOCAL ps:PAINTSTRUCT
            LOCAL tnid:NOTIFYICONDATA
; обробка повідомлення про ініціалізацію діалогу
            .IF uMsg == WM_INITDIALOG
; створення рисунка з фотографією
INVOKE CreateWindowEx, WS_EX_STATICEDGE, ADDR statClass, 0,
WS_CHILD or SS_BITMAP, 0, 0, 10, 10, hWnd, 65535, hInstance, 0
            mov hFotoImage, eax
            INVOKE ShowWindow, hFotoImage, SW_HIDE
            INVOKE LoadBitmap, hInstance, 750
invoke SendMessage, hFotoImage, STM_SETIMAGE, IMAGE_BITMAP, eax
; створення рисунка з іконкою
INVOKE CreateWindowEx, WS_EX_LEFT, ADDR statClass, NULL,
WS_CHILD or WS_VISIBLE or SS_ICON, 10, 10, 32, 32, hWnd, 65534,
hInstance, NULL
INVOKE SendMessage, eax, STM_SETIMAGE, IMAGE_ICON, hIcon
; створення іконки програми в правому нижньому кутку екрана
            mov tnid.cbSize, SIZEOF NOTIFYICONDATA
            mov eax, hWnd
            mov tnid.hwnd, eax
            mov tnid.uID, 0
            mov tnid.uFlags, NIF_MESSAGE or NIF_ICON or NIF_TIP
            mov tnid.uCallbackMessage, 0
            mov eax, hIcon
            mov tnid.hIcon, eax
            mov tnid.szTip[0], 0
            INVOKE Shell_NotifyIcon, NIM_ADD, ADDR tnid
; установка таймера для WM_TIMER кожні 10 мс (для анімації квадратів)
            INVOKE SetTimer, hWnd, 0, 10, NULL
; обробка повідомлення таймера
            .ELSEIF uMsg == WM_TIMER
                    .If Mode == ShowSquares
; періодичне перерисування вікна (для анімації квадратів)
```

```
                INVOKE InvalidateRect, hWnd, NULL, TRUE
                .ENDIF
; обробка повідомлення про рисування
        .ELSEIF uMsg == WM_PAINT
                INVOKE BeginPaint, hWnd, ADDR ps
                mov hDC, eax
                INVOKE DrawSquares, hWnd, hDC
                INVOKE DrawAvtor, hWnd, hDC
                INVOKE DrawTask, hWnd, hDC
                INVOKE EndPaint, hWnd, ADDR ps
; обробка повідомлення про закриття діалогу
        .ELSEIF uMsg==WM_DESTROY
; видалення таймера
                INVOKE KillTimer, hWnd, 0
; видалення іконки програми в правому нижньому кутку екрана
                mov tnid.cbSize, SIZEOF NOTIFYICONDATA
                mov eax, hWnd
                mov tnid.hwnd, eax
                mov tnid.uID, 0
                INVOKE Shell_NotifyIcon, NIM_DELETE, ADDR tnid
                INVOKE PostQuitMessage, 0
; обробка повідомлення про команду користувача
        .ELSEIF uMsg==WM_COMMAND
                mov eax, wParam
; обробка повідомлення про вибір меню
                .IF lParam==0
                        .IF ax==IDM_SQUARES
                INVOKE ShowWindow, hFotoImage, SW_HIDE
                mov Mode, ShowSquares
                        .ELSEIF ax==IDM_AVTOR
                INVOKE ShowWindow, hFotoImage, SW_HIDE
                        mov Mode, ShowAvtor
                        .ELSEIF ax==IDM_FOTO
                INVOKE ShowWindow, hFotoImage, SW_SHOW
                        mov Mode, ShowFoto
                        .ELSEIF ax==IDM_TASK
                INVOKE ShowWindow, hFotoImage, SW_HIDE
                        mov Mode, ShowTask
                        .ELSEIF ax==IDM_EXIT
                INVOKE DestroyWindow, hWnd
                        .ENDIF
                INVOKE InvalidateRect, hWnd, NULL, TRUE
; обробка повідомлення про натиснення на кнопку OK
                .ELSE
                        mov edx, wParam
                        shr edx, 16
```

```
                        .IF dx==BN_CLICKED
                                .IF ax==IDC_BUTTON
                                INVOKE DestroyWindow, hWnd
                                .ENDIF
                        .ENDIF
                .ENDIF
; обробка решти повідомлень за умовчанням
        .ELSE
                INVOKE DefWindowProc, hWnd, uMsg, wParam, lParam
                ret
        .ENDIF
        xor     eax, eax
        ret
    WndProc endp
start:
; реєстрація класу діалогу
        mov wc.cbSize, SIZEOF WNDCLASSEX
        mov wc.style, CS_HREDRAW or CS_VREDRAW
        mov wc.lpfnWndProc, OFFSET WndProc
        mov wc.cbClsExtra, NULL
        mov wc.cbWndExtra, DLGWINDOWEXTRA
        INVOKE GetModuleHandle, NULL
        mov hInstance, eax
        mov wc.hInstance, eax
        mov wc.hbrBackground, COLOR_WINDOW+1
        mov wc.lpszMenuName, OFFSET MenuName
        mov wc.lpszClassName, OFFSET ClassName
        INVOKE LoadIcon, hInstance, 500
        mov hIcon, eax
        mov wc.hIcon, eax
        mov wc.hIconSm, eax
       INVOKE LoadCursor, NULL, IDC_CROSS
        mov wc.hCursor, eax
        INVOKE RegisterClassEx, addr wc
INVOKE CreateDialogParam, hInstance, ADDR DlgName, NULL, WndProc, 0
        mov hDlg, eax
        INVOKE ShowWindow, hDlg, SW_SHOWNORMAL
        INVOKE UpdateWindow, hDlg
; цикл обробки повідомлень
        .WHILE TRUE
                INVOKE GetMessage, ADDR msg, NULL, 0, 0
                .BREAK .IF (!eax)
                INVOKE TranslateMessage, ADDR msg
                INVOKE DispatchMessage, ADDR msg
        .ENDW
        mov eax, msg.wParam
```

```
        INVOKE ExitProcess, eax
    end start
```

3.4. Отримання IP-адреси комп'ютера

У лістингу 3.7 наведено програму отримання IP-адреси.

Лістинг 3.7. Отримання IP-адреси комп'ютера:

```
    .386                    ; директива визначення типу мікропроцесора
.model flat,stdcall ; завдання лінійної моделі пам'яті та угоди ОС Windows
     option casemap:none        ; відмінність малих та великих літер
     include \masm32\include\windows.inc   ; файли структур, констант …
     include \masm32\macros\macros.asm
     uselib user32, kernel32, wsock32, fpu, gdi32
WndProc PROTO :DWORD,:DWORD,:DWORD,:DWORD

.data                               ; директива визначення даних
   dlgname     db "WINSOCK",0
   szTitle     db "Yours IP address",0
   wsaError    db "Error initializing winsock!",13,10
   szName      db "Yours Computer Name: %s",0
   szFont      db "MS Sans Serif",0
About_string db "Програма відображає назву ПК та IP адрес ПК",10,13
  str1 db "Виконав …",10,13
  str2 db " e-mail  …",0
  str3 db "INFORMATION ABOUT PROGRAM",0
.data?                              ; директива невизначених даних
     wsa WSADATA <?>
     hStatic    dd ?
     hFont      dd ?
     hInstance  dd ?
     buffer     db 24 dup (?)
     buffer2    db 128 dup (?)
.code                               ; директива початку сегмента команд
start:                              ; мітка початку програми
invoke MessageBox,NULL,ADDR About_string,addr str3,\
   MB_ICONINFORMATION  ; виведення повідомлення про автора
     invoke GetModuleHandle, NULL   ; отримання дескриптора програми
     mov hInstance, eax             ; збереження дескриптора програми
     invoke WSAStartup,101h,addr wsa ; утримання версії Windows Sockets
.if eax == NULL
invoke DialogBoxParam,hInstance,ADDR dlgname,0,ADDR WndProc,0
                              ; функція створення модального діалогового вікна
          invoke ExitProcess,0
.endif
```

```
invoke MessageBox,NULL,offset wsaError,offset szTitle,\
 MB_OK + MB_ICONSTOP ; виведення повідомлення про помилку
                                                    ; у результаті створення вікна
      invoke ExitProcess,1

WndProc proc hWin:DWORD,uMsg:DWORD,wParam:DWORD,\
                                                    lParam:DWORD
.if uMsg == WM_INITDIALOG       ; повідомлення посилання віконній
       ; процедурі діалогового вікна безпосередньо перед тим, як воно буде
                                                    ; відображатися на екрані
 invoke LoadIcon,hInstance,101   ; завантаження ресурсу піктограми
 invoke SendMessage,hWin,WM_SETICON,TRUE,eax ; функція відсилання
                              ; повідомлення вікну, хендл вікна, тип повідомлення
invoke GetDlgItem,hWin,2000  ; функція відновлення хендла контролу
                                                    ; у вказаному діалоговому вікні
          mov hStatic,eax
invoke gethostname,offset buffer,sizeof buffer  ; функція отримання
                                                    ; імені комп'ютера
invoke wsprintf,addr buffer2,addr szName,addr buffer
   invoke SetDlgItemText,hWin,3000,addr buffer2      ; функція встановлення
                                ; тексту елемента керування в діалоговому вікні
invoke gethostbyname,addr buffer  ; функція отримання IP-адреси
         mov eax,[eax+12]         ; отримуємо покажчик
         mov eax,[eax]            ; отримуємо покажчик на покажчик IP
mov eax,[eax]  ; отримання покажчика на рядок IP
   invoke inet_ntoa,eax              ; функція перетворення
   invoke SetDlgItemText,hWin,2000,eax  ; функція встановлення тексту
                                     ; елемента управління в діалоговому вікні
              invoke WSACleanup
              xor eax,eax
       ret
.elseif uMsg == WM_CTLCOLORSTATIC
        mov eax,lParam
        .if eax == hStatic
 invoke CreateFont,16,16,0,0,400,0,0,0,OEM_CHARSET,\
  OUT_TT_PRECIS,CLIP_DEFAULT_PRECIS,\
    DEFAULT_QUALITY,DEFAULT_PITCH or FF_SWISS,\
     ADDR szFont
         mov hFont,eax
         invoke SelectObject, wParam, hFont    ; вибір у контекст
         invoke GetSysColor, COLOR_MENU  ; установлення кольору меню
         invoke SetBkColor, wParam, eax      ; установлення фону за текстом
         invoke SetTextColor,wParam,Blue    ; установлення кольору тексту
         invoke GetStockObject, HOLLOW_BRUSH    ; читання описувача
               ret
        .endif
```

```
.elseif uMsg == WM_CLOSE  ; якщо є повідомлення про закриття вікна
    invoke DeleteObject,hFont  ; видаляє Handl з пам'яті та звільняє
ресурси
    invoke EndDialog,hWin,0 ; функція знищення модального діал. вікна
        xor eax,eax              ; підготування до закінчення
        ret                      ; повернення з процедури
.endif
      xor eax,eax                ; підготування до закінчення
      ret                        ; повернення з процедури
      WndProc endp               ; закінчення процедури WndProc
      end start                  ; закінчення програми з ім'ям start
```

Лістинг 3.8. Файл ресурсів програми отримання IP-адреси комп'ютера**:**

```
      #include "\masm32\include\resource.h"
      WINSOCK DIALOG  0, 0, 175,85
STYLE DS_CENTER | WS_BORDER | WS_CAPTION | WS_MINIMIZEBOX |
 WS_SYSMENU  | WS_VISIBLE | WS_OVERLAPPED | DS_MODALFRAME |
  DS_3DLOOK
      CAPTION " IP address"
      FONT 8, "Arial"
BEGIN
   GROUPBOX   "&Main",3000,8,45,160,35
   CTEXT      "Static",2000,12,58,145,12,SS_CENTERIMAGE | SS_SUNKEN
END
```

На рисунку 3.4 наведено результат виконання програми отримання IP-адреси комп'ютера.

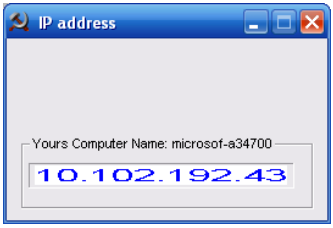


Рисунок 3.4 – Результат виконання програми отримання IP-адреси комп'ютера

3.5. Анімація іконки

Для створення діалогового вікна використано програму ResEd роботи з ресурсами. Після створення вікна в цю програму внесені зміни для визначення імен ікон (лістинг 3.9).

Лістинг 3.9. Файл ресурсів програми анімації іконки:

```
#define RT_MANIFEST 24
#define IDI_ICON0 50
#define IDI_ICON1 51
#define IDI_ICON2 52
#define IDI_ICON3 53
#define IDI_ICON4 54
#define IDI_ICON5 55
#define IDI_ICON6 56
#define IDI_ICON7 57
#define IDI_ICON8 58
#define IDI_ICON9 59
#define IDD_DEMODLG 80
#include "/masm32/include/resource.h"

IDI_ICON0 ICON DISCARDABLE "Mt11.ico"
IDI_ICON1 ICON DISCARDABLE "Mt12.ico"
IDI_ICON2 ICON DISCARDABLE "Mt13.ico"
IDI_ICON3 ICON DISCARDABLE "Mt14.ico"
IDI_ICON4 ICON DISCARDABLE "Mt15.ico"
IDI_ICON5 ICON DISCARDABLE "Mt21.ico"
IDI_ICON6 ICON DISCARDABLE "Mt22.ico"
IDI_ICON7 ICON DISCARDABLE "Mt23.ico"
IDI_ICON8 ICON DISCARDABLE "Mt24.ico"
IDI_ICON9 ICON DISCARDABLE "Mt25.ico"

IDD_DEMODLG DIALOGEX 0,0,190,55
CAPTION "Иконка"
FONT 10,"MS Sans Serif",0,0,0
STYLE
WS_VISIBLE|WS_CAPTION|WS_SYSMENU|DS_CENTER|DS_SYSMODAL
BEGIN
  CONTROL "",-
1,"Button",WS_CHILDWINDOW|WS_VISIBLE|BS_FLAT|BS_CENTER|BS_G
ROUPBOX,126,-1,60,25
  CONTROL "Анимация иконки!",-
1,"Static",WS_CHILDWINDOW|WS_VISIBLE|WS_GROUP|SS_CENTER,33,
33,126,12
  CONTROL
"Exit",IDCANCEL,"Button",WS_CHILDWINDOW|WS_VISIBLE|WS_TABSTO
P|BS_FLAT,132,6,50,15,WS_EX_DLGMODALFRAME
END
```

Програма анімації наведена в лістингу 3.10.

Лістинг 3.10. Программа анімації:

```
        .386
INCLUDE \Masm32\Include\masm32rt.inc
        TIMER_ELAPSE EQU 100
        IDI_ICON0    EQU 50
        IDI_ICON1    EQU 51
        IDI_ICON2    EQU 52
        IDI_ICON3    EQU 53
        IDI_ICON4    EQU 54
        IDI_ICON5    EQU 55
        IDI_ICON6    EQU 56
        IDI_ICON7    EQU 57
        IDI_ICON8    EQU 58
        IDI_ICON9    EQU 59

        IDD_DEMODLG  EQU 80
        IDT_TIMER0   EQU 90
        IFNDEF       SC_DRAGMOVE
        SC_DRAGMOVE  EQU 0F012h
        ENDIF

.DATA?
        hInstance dd ?
        uFrameCnt dd ?
        hIcons    dd 10 dup(?)
        nid       NOTIFYICONDATA <>

.CODE
DlgProc PROC    hWnd:HWND,uMsg:UINT,wParam:WPARAM,\
                        lParam:LPARAM
        mov     eax,uMsg
        cmp     eax,WM_TIMER
        jz      WmTimer
      cmp     eax,WM_LBUTTONDOWN
      jz      WmLButtonDown
        cmp     eax,WM_COMMAND
        jz      WmCommand
      cmp     eax,WM_INITDIALOG
      jz      WmInitDialog
        cmp     eax,WM_CLOSE
        jz      WmClose

UnProcessed:
        xor     eax,eax
DlgExit:
        ret
WmTimer:
```

```
        mov     eax,uFrameCnt
        xor     edx,edx
        inc     eax
        cmp     eax,25
        mov     ecx,eax
        jb      WmTimer1

        mov     eax,edx
        mov     ecx,edx
        jmp short WmTimer4
WmTimer1:
        sub     eax,5
        jc      WmTimer2
        sub     eax,5
        jc      WmTimer2
        sub     eax,5
        jc      WmTimer2
        add     edx,4*5
        sub     eax,5
        jnc     WmTimer3
WmTimer2:
        add     eax,5
WmTimer3:
        shl     eax,2
WmTimer4:
        mov     eax,hIcons[eax+edx]
        mov     uFrameCnt,ecx
        mov     nid.hIcon,eax
        INVOKE  SendMessage,hWnd,WM_SETICON,ICON_BIG,eax
        INVOKE  Shell_NotifyIcon,NIM_MODIFY,offset nid
Processed:
        mov     eax,1
        jmp     DlgExit
WmLButtonDown:
        INVOKE  ReleaseCapture
        INVOKE  SendMessage,hWnd,WM_SYSCOMMAND,SC_DRAGMOVE,0
        jmp     Processed

WmInitDialog:
        push    esi
        push    ebx
        push    edi
        mov     esi,offset nid
        mov     eax,hIcons
        mov     ebx,hWnd
        xor     edi,edi
```

```
        mov     [esi].NOTIFYICONDATA.hIcon,eax
        INVOKE  SendMessage,ebx,WM_SETICON,ICON_BIG,eax
        mov     [esi].NOTIFYICONDATA.cbSize,sizeof NOTIFYICONDATA
        mov     [esi].NOTIFYICONDATA.hwnd,ebx
        mov     [esi].NOTIFYICONDATA.uID,IDT_TIMER0
        mov     [esi].NOTIFYICONDATA.uFlags,NIF_ICON
        mov     [esi].NOTIFYICONDATA.uCallbackMessage,edi
        INVOKE  Shell_NotifyIcon,NIM_ADD,esi
        INVOKE  SetTimer,ebx,IDT_TIMER0,TIMER_ELAPSE,edi
        pop     edi
        pop     ebx
        pop     esi
        jmp     Processed

WmCommand:
        cmp     wParam,BN_CLICKED SHL 16 OR IDCANCEL
        jnz     Processed

WmClose:
        INVOKE  KillTimer,hWnd,IDT_TIMER0
        INVOKE  Shell_NotifyIcon,NIM_DELETE,offset nid
        INVOKE  EndDialog,hWnd,0
        jmp     Processed
DlgProc ENDP


_main   PROC
        xor     edi,edi
        push    edi
        push    sizeof INITCOMMONCONTROLSEX
        INVOKE  InitCommonControlsEx,esp
        INVOKE  GetModuleHandle,edi
        mov     hInstance,eax
        mov     esi,IDI_ICON9
        mov     ebx,4*(IDI_ICON9-IDI_ICON0)
LdIcon: INVOKE  LoadIcon,hInstance,esi
        mov     hIcons[ebx],eax
        dec     esi
        sub     ebx,4
        jnc     LdIcon
  INVOKE  DialogBoxParam,hInstance,IDD_DEMODLG,edi,DlgProc,edi
  INVOKE  ExitProcess,edi
_main   ENDP
        END     _main
```

Результат роботи програми наведено на рисунку 3.5.

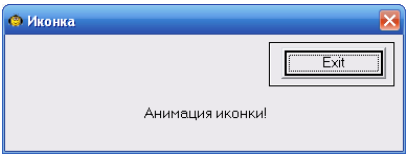


Рисунок 3.5 – Результат роботи програми

3.6. Виведення азбуки Морзе на системний динамік

Азбука Морзе (*телеграфна азбука*) – назване за ім'ям розробника Семюела Морзе відтворення графічних знаків комбінацією крапок і тире. За одиницю часу береться тривалість однієї точки. Тривалість тире дорівнює трьом точкам. Пауза між елементами одного знака – одна точка, між знаками в слові – 3 точки, між словами – 7 точок.

Програма перетворює в сигнали тільки кирилічну азбуку.

При отриманні повідомлення про створення вікна виконуються дії:

– викликається API-функція AnimateWindow, яка прорисовує вікно зверху вниз, тобто використовується анімація;

– викликається API-функція створення області для введення тексту (місце введення слів для генеруваня азбуки Морзе) CreateWindowEx;

– викликається API-функція створення кнопки генеруваня азбуки Морзе (button) CreateWindowEx.

API-функція SetWindowText встановлює константний рядок ("сос сос сос") в поле введення тексту для генерування сигнального коду. Даний рядок зберігається в змінній TestString (використовується як параметр функції).

При отриманні повідомлення про введення тексту у відповідне поле виконуються дії:

– очищається буфер для збереження введених символів [2, 3];

– викликається API-функція GetWindowText, яка зберігає введений рядок у змінну buffer;

– запам’ятовується адреса початкових символів константного рядка та введеного рядка (buffer);

– у циклі за допомогою операцій repnz scasb порівнюється кожен елемент введеного рядка з кожним елементом алфавіту;

– якщо елемент був знайдений, то зменшується регістр, де знаходиться адреса рядка алфавіту на одиницю (щоб адреса вказувала на

той елемент, який був знайдений, а не на наступний). Якщо не було знайдено символ, то цикл завершується ;

– залежно від того, який символ був знайдений, виводиться звуковий сигнал. За це відповідають макроси (якщо символ дорівнює пропуску – робиться пауза з інтервалом 700 мс, пропуск між символами – 300 мс, між сигналами – 100 мс);

Лістинг 3.11. Файл ресурсів програми виведення азбуки Морзе:

```
#define IDM_HELLO 1
#define IDM_GETTEXT 3
#define IDM_EXIT 4
#define IDM_ZAVD 5
#define IDM_INFO 6
#define IDI_ICON 22
#define IDM_TABL 7
IDI_ICON ICON DISCARDABLE MOVEABLE LOADONCALL  "Mac OS X 10.2 _Jaguar_.ico"
FirstMenu MENU
{
 POPUP "&Головне Меню"
      {
       MENUITEM "Відображення пробного тексту",IDM_HELLO
       MENUITEM "Генерування азбуки морзе", IDM_GETTEXT
       MENUITEM SEPARATOR
      }
POPUP "ДОВІДКА"
     {
      MENUITEM "Інформація про завдання", IDM_ZAVD
      MENUITEM "Таблиця морзе-кодів", IDM_TABL
       POPUP "Автор"{
          MENUITEM "Інформація про автора",IDM_INFO
        }
      MENUITEM SEPARATOR
     }
MENUITEM "ВИХІД",IDM_EXIT
}
```

Програма виведення азбуки Морзе наведена в лістингу 3.12.

Лістинг 3.12. Програма виведення азбуки Морзе:

```
.686                    ; директива визначення типу мікропроцесора
.model flat,stdcall  ; завдання лінійної моделі пам’яті та угоди ОС Windows
option casemap:none  ; відмінність малих та великих літер
WinMain proto :DWORD,:DWORD,:DWORD,:DWORD ; прототип процедури
include \masm32\include\windows.inc ; файли структур, констант …
```

```
include \masm32\macros\macros.asm
uselib masm32, user32, kernel32, gdi32
  @ MACRO b0,b1,b2,b3
  b0
   b1
    b2
     b3
  ENDM
Macro_Beep4 macro arc1,arc2,arc3,arc4
  invoke Beep,01111,arc1 ;; частота звука та довжина
  invoke Beep,0,100          ;; пауза 100 мс
  invoke Beep,01111,arc2
  invoke Beep,0,100
  invoke Beep,01111,arc3
  invoke Beep,0,100
  invoke Beep,01111,arc4
  invoke Beep,0,300
endm
Macro_Beep3 macro arc1,arc2,arc3
  invoke Beep,01111,arc1
  invoke Beep,0,100
  invoke Beep,01111,arc2
  invoke Beep,0,100
  invoke Beep,01111,arc3
  invoke Beep,0,300
endm
Macro_Beep2 macro arc1,arc2
  invoke Beep,01111,arc1
  invoke Beep,0,100
  invoke Beep,01111,arc2
  invoke Beep,0,300
endm
Macro_Beep1 macro arc1
  invoke Beep,01111,arc1
  invoke Beep,0,300
endm
.data                              ; директива визначення даних
constan_str  db  'абвгдежзийклмнопрстуфхцчшщъыьэюя  АБВГДЕЖЗИЙКЛ
МНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'
    cons_kol dd 65
    kor_sg dd 100 ; довжина  точки
    dl_sg  dd 300  ;  довжина  тире
    ClassName db "SimpleWinClass",0
AppName   db "Курсова на тему 'Виведення азбуки Морзе на системний
динамік'",0
    MenuName db "FirstMenu",0
```

```
    ButtonClassName db "button",0
    ButtonText db "ГЕНЕРУВАННЯ АЗБУКИ МОРЗЕ",0
    EditClassName db "edit",0
    TestString db "сос сос сос",0
    str_exit db 'Ви впевнені,що бажаєте вийти із програми ?',0
    zag_exit db 'Вихід з програми',0
    str_avtor db "Автор програми:  ",10,13,
            "Місце навчання : НТУ 'ХПІ'",10,13,
            "Факультет : Комп'ютерні та Інформаційні технології",10,13,0
zag_avtor db "Інформація про автора",0
zavd db "Створити програму, яка б виводила сигнали азбуки Морзе на
системний динамік",0
zag_zavd db "Інформація про завдання",0
tabl_morze db "а   *- б   -*** в   *--  г --*   д   -**",0dh,0ah,
            "е   *     ж  ***- з   --** и    ** й  *---",0dh,0ah,
            "к   -*-   л  -**  м   --    н   -*  о   ---",0dh,0ah,
            "п  *--*   р  *-* с   ***    т   -   у  **-",0dh,0ah,
            "ф  **-*   х  **** ц  -*-* ч  ---* ш  ----",0dh,0ah,
            "щ  --*-  ы  -*--  ь   -**- ю **-- я  *-*-",0dh,0ah,0
.data?                          ; директива невизначених даних
  hInstance HINSTANCE ?
  CommandLine LPSTR ?
  hwndButton HWND ?
  hwndEdit HWND ?
  buffer db 512 dup(?)
  len1 equ ($-buffer)/type buffer
.const
    ButtonID equ 1
    EditID   equ 2
    IDM_HELLO equ 1
    IDM_GETTEXT equ 3
    IDM_EXIT equ 4
    IDI_ICON  equ 22
    IDM_ZAVD equ 5
    IDM_INFO equ 6
    IDM_TABL equ 7
.code                           ; директива початку сегмента команд
 start:                         ; мітка початку програми
invoke GetModuleHandle, NULL ; отримання дескриптора програми
            mov    hInstance,eax            ; збереження дескриптора програми
            invoke GetCommandLine
            mov CommandLine,eax
invoke WinMain, hInstance,NULL,CommandLine, SW_SHOWDEFAULT
invoke ExitProcess,eax ; повернення керування ОС та визволення ресурсів

WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,\
```

```
  CmdShow:DWORD
LOCAL wc:WNDCLASSEX   ; резервування стека під структуру
LOCAL msg:MSG         ; резервування стека під структуру MSG
LOCAL hwnd:HWND       ;  резервування стека під хендл програми
@<mov wc.cbSize,SIZEOF WNDCLASSEX>,<mov wc.cbClsExtra,0>
@<mov wc.style,CS_HREDRAW or CS_VREDRAW >,<mov
wc.cbWndExtra,0>
@<mov wc.lpfnWndProc,OFFSET WndProc>,<push hInst>,<pop wc.hInstance
>
@<mov wc.hbrBackground,COLOR_WINDOW+1>,<mov wc.lpszMenuName,
OFFSET MenuName>
        mov   wc.lpszClassName,OFFSET ClassName  ; ім'я класу
invoke LoadIcon,hInstance,IDI_ICON      ; ресурс піктограми
@< mov   wc.hIcon,eax >,< mov   wc.hIconSm,eax >
        invoke LoadCursor,NULL,IDC_ARROW    ; ресурс
        mov   wc.hCursor,eax     ; дескриптор курсора
invoke RegisterClassEx, addr wc  ; функція реєстрації класу вікна
invoke CreateWindowEx, \         ; функція створення вікна за зразком
 WS_EX_CLIENTEDGE,ADDR ClassName, \    ; стиль та адреса імені класу
  ADDR AppName,\                       ; адреса імені вікна
  WS_OVERLAPPEDWINDOW xor WS_THICKFRAME,\      ; базовий стиль
   0, 0,550,300,\      ; x0,y0, горизонтальна та вертикальна координати вікна
     NULL,NULL,hInst,NULL
        mov   hwnd,eax
        invoke ShowWindow, hwnd,SW_SHOWNORMAL ; видимість вікна
   invoke UpdateWindow, hwnd
        .WHILE TRUE                  ; поки істинно, то
         invoke GetMessage, ADDR msg,NULL,0,0
      .BREAK .IF (!eax)
invoke TranslateMessage, ADDR msg
invoke DispatchMessage, ADDR msg ; відправка на обслуговування
        .ENDW
        mov     eax,msg.wParam
        ret                          ; повернення з процедури
WinMain endp                         ; закінчення процедури з ім'ям WinMain

WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM,\
          lParam:LPARAM
    LOCAL hDC:HDC
    LOCAL PaintStruct:PAINTSTRUCT
    LOCAL rect:RECT
.IF uMsg==WM_CREATE          ; якщо є повідомлення про створення вікна
invoke AnimateWindow,hWnd,2000,AW_ACTIVATE+AW_VER_POSITIVE
                                                          ; анімація
   invoke EndPaint,hWnd,ADDR PaintStruct
   invoke ReleaseDC, hWnd, hDC
```

```
invoke CreateWindowEx,WS_EX_CLIENTEDGE, ADDR EditClassName,0,\
                                        ; створення області для введення слів
     WS_CHILD or WS_VISIBLE or WS_BORDER or ES_LEFT or\
       ES_AUTOHSCROLL, 115,130,300,25,hWnd,EditID,hInstance,NULL
            mov  hwndEdit,eax
invoke SetFocus, hwndEdit
invoke CreateWindowEx,NULL, ADDR ButtonClassName,ADDR ButtonText,\
 WS_CHILD     or     WS_VISIBLE     or     BS_DEFPUSHBUTTON     or
WS_EX_TRANSPARENT or  WS_THICKFRAME,\
           140,185,250,40,hWnd,ButtonID,hInstance,NULL ; кнопка
            mov  hwndButton,eax
.ELSEIF uMsg==WM_COMMAND          ; якщо виконується якась команда
            mov eax,wParam
            .IF lParam==0
               .IF ax==IDM_HELLO
invoke SetWindowText,hwndEdit,ADDR TestString  ; встановлення тексту
       .ELSEIF  ax==IDM_GETTEXT
             mov ecx,len1
         .while(ecx!=0)
             mov buffer[ecx],0       ; очистка буфера
             dec ecx
         .endw
invoke GetWindowText,hwndEdit,ADDR buffer,len1; збереження тексту
@<lea edi,constan_str>,<mov edx,edi>,<lea esi,buffer>,<mov ebx,len1>
    .while (ebx!=0)
      lea edi,constan_str ; edi-адреса початку конст. рядка
      mov al,[esi]            ; al – символ, який порівнюється
      mov ecx,cons_kol
      cld
      repnz scasb          ; порівнювати поки символ не буде знайдений
      jmp m5
m1:
       dec edi
       lea edx,constan_str
       sub edi,edx
    .if((edi==0)||(edi==33))                    ; якщо а
      Macro_Beep2 [kor_sg],[dl_sg]                    ; *-
    .elseif((edi==1)||(edi==34))                ; якщо б
      Macro_Beep4 [dl_sg],[kor_sg],[kor_sg],[kor_sg]        ; -***
    .elseif((edi==2)||(edi==35))                ; якщо в
      Macro_Beep3 [kor_sg],[dl_sg],[dl_sg]              ; *--
    .elseif((edi==3)||(edi==36))                ; якщо г
      Macro_Beep3 [dl_sg],[dl_sg],[kor_sg]              ; --*
    .elseif((edi==4)||(edi==37))                ; якщо д
      Macro_Beep3 [dl_sg],[kor_sg],[kor_sg]             ; -**
    .elseif((edi==5)||(edi==38))                ; якщо е
```

```
  Macro_Beep1 [kor_sg]                          ; *
.elseif((edi==6)||(edi==39))                    ; якщо ж
  Macro_Beep4 [kor_sg],[kor_sg],[kor_sg],[dl_sg]        ; ***-
.elseif((edi==7)||(edi==40))                    ; якщо з
  Macro_Beep4 [dl_sg],[dl_sg],[kor_sg],[kor_sg]         ; --**
.elseif((edi==8)||(edi==41))                    ; якщо и
    Macro_Beep2 [kor_sg],[kor_sg]                   ;**
.elseif((edi==9)||(edi==42))                    ; якщо й
    Macro_Beep4 [kor_sg],[dl_sg],[dl_sg],[dl_sg]          ;* ---
.elseif((edi==10)||(edi==43))                   ; якщо к
    Macro_Beep3 [dl_sg],[kor_sg],[dl_sg]              ; -*-
.elseif((edi==11)||(edi==44))                   ; якщо л
    Macro_Beep4 [kor_sg],[dl_sg],[kor_sg],[kor_sg]        ; *-**
.elseif((edi==12)||(edi==45))                   ; якщо м
    Macro_Beep2 [dl_sg],[dl_sg]                     ; --
.elseif((edi==13)||(edi==46))                   ; якщо н
    Macro_Beep2 [dl_sg],[kor_sg]                    ; -*
.elseif((edi==14)||(edi==47))                   ; якщо о
    Macro_Beep3 [dl_sg],[dl_sg],[dl_sg]                   ; ---
.elseif((edi==15)||(edi==48))                    ; якщо п
  Macro_Beep4 [kor_sg],[dl_sg],[dl_sg],[kor_sg]         ; *--*
.elseif((edi==16)||(edi==49))                   ; якщо р
    Macro_Beep3 [kor_sg],[dl_sg],[kor_sg]             ; *-*
.elseif((edi==17)||(edi==50))                    ; якщо с
    Macro_Beep3 [kor_sg],[kor_sg],[kor_sg]              ; ***
.elseif((edi==18)||(edi==51))                   ; якщо т
    Macro_Beep1 [dl_sg]                         ; -
.elseif((edi==19)||(edi==52))                    ; якщо у
    Macro_Beep3 [kor_sg],[kor_sg],[dl_sg]             ; **-
.elseif((edi==20)||(edi==53))                    ; якщо ф
    Macro_Beep4 [kor_sg],[kor_sg],[dl_sg],[kor_sg]        ; **-*
.elseif((edi==21)||(edi==54))                   ; якщо х
    Macro_Beep4 [kor_sg],[kor_sg],[kor_sg],[kor_sg]       ; ****
.elseif((edi==22)||(edi==55))                   ; якщо ц
    Macro_Beep4 [dl_sg],[kor_sg],[dl_sg],[kor_sg]         ; -*-*
.elseif((edi==23)||(edi==56))                   ; якщо ч
    Macro_Beep4 [dl_sg],[dl_sg],[dl_sg],[kor_sg]          ; ---*
.elseif((edi==24)||(edi==57))                   ; якщо ш
    Macro_Beep4 [dl_sg],[dl_sg],[dl_sg],[dl_sg]           ; ----
.elseif((edi==25)||(edi==58))                   ; якщо щ
    Macro_Beep4 [dl_sg],[dl_sg],[kor_sg],[dl_sg]          ; --*-
.elseif((edi==26)||(edi==59))                    ; якщо ъ
    Macro_Beep4 [dl_sg],[kor_sg],[kor_sg],[dl_sg]     ; -**-
.elseif((edi==27)||(edi==60))                   ; якщо ы
    Macro_Beep4 [dl_sg],[kor_sg],[dl_sg],[dl_sg]          ; -*--
.elseif((edi==28)||(edi==61))                   ; якщо ь
```

```
      Macro_Beep4 [dl_sg],[kor_sg],[kor_sg],[dl_sg]         ; -**-
   .elseif((edi==29)||(edi==62))                    ; якщо э
      Macro_Beep4 [kor_sg],[dl_sg],[kor_sg],[kor_sg]  ; *-**
   .elseif((edi==30)||(edi==63))                    ; якщо ю
      Macro_Beep4 [kor_sg],[kor_sg],[dl_sg],[dl_sg]         ; **--
   .elseif((edi==31)||(edi==64))                    ; якщо я
     Macro_Beep4 [kor_sg],[dl_sg],[kor_sg],[dl_sg]         ; *-*-
   .elseif(edi==32)                                 ; якщо пробіл
      invoke Beep,0,700
   .endif
      jmp m4
m5:   jz m1
      jnz m4
m4:
       inc esi
       dec ebx
   .endw
   .ELSEIF ax==IDM_ZAVD
invoke MessageBox,NULL,ADDR zavd,ADDR
zag_zavd,MB_ICONINFORMATION ; виведення інформації про завдання
   .ELSEIF ax==IDM_TABL
invoke MessageBox,NULL,ADDR tabl_morze,ADDR zag_avtor,
MB_ICONINFORMATION  ; виведення таблиці
   .ELSEIF ax==IDM_INFO
invoke MessageBox,NULL,ADDR str_avtor,ADDR
zag_avtor,MB_ICONINFORMATION  ; виведення інформації про автора
   .ELSE
invoke MessageBox,NULL,ADDR str_exit,ADDR zag_exit,MB_OKCANCEL or
MB_ICONWARNING
      .if(eax==1)
invoke DestroyWindow,hWnd
      .endif
   .ENDIF
            .ELSE
                  .IF ax==ButtonID
                  shr eax,16
                  .IF ax==BN_CLICKED
invoke SendMessage,hWnd,WM_COMMAND,IDM_GETTEXT,0
                  .ENDIF
                  .ENDIF
         .ENDIF
.ELSEIF uMsg==WM_DESTROY ; є повідомлення про знищення вікна
invoke PostQuitMessage,NULL    ; передача повідомлення про знищення
         .ELSE
invoke DefWindowProc,hWnd,uMsg,wParam,lParam
                  ret
```

```
        .ENDIF
        xor    eax,eax          ; підготування до закінчення
        ret                     ; повернення з процедури
WndProc endp                    ; закінчення процедури WndProc
end start                       ; закінчення програми з ім'ям start
```

Лістинг 3.13. Командний bat-файл:

```
ml /c /coff "MorZe.asm"
rc "MorZe.rc"
link /SUBSYSTEM:windows "MorZe.obj"  "MorZe.res"
del  MorZe.obj MorZe.res
pause
start MorZe.exe
```

Результат роботи програми наведено на рисунку 3.6.

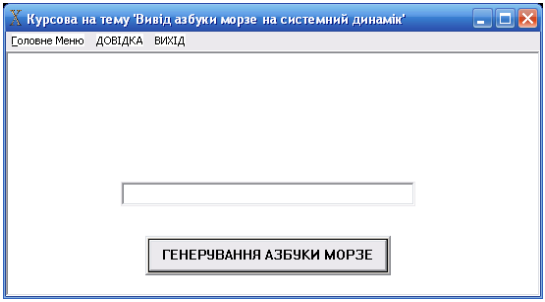


Рисунок 3.6 – Результат роботи програми

4. СТРУКТУРА ПРОЕКТУ

При загальному характері запису питань пояснювальної записки конкретний вид розроблювальних засобів залежить від варіанта завдання. Тому в оформленні пояснювальної записки формулювання розроблених питань може бути конкретизовано.

Склад і зміст курсового проекту мають відповідати затвердженій темі та отриманому завданню. Зміст курсового проекту розробляється студентом разом з керівником.

Обсяг курсового проекту орієнтовно визначається складом питань, що підлягають розробці, а також переліком обов'язкових графічних робіт, зазначених у завданні. Обсяг пояснювальної записки повинен бути не менше 15 сторінок.

4.1. Вимоги до курсового проекту

Під час виконання курсового проекту з навчальної дисципліни «Системне програмування», в якій вивчається мова асемблера та його використання пропонується дотримуватися таких умов:

– розв'язати задачу за допомогою тільки мови **асемблера masm32**;

– організувати введення-виведення повідомлень у діалогові (якщо можливо) **вікна** екрана з елементами **меню** під ОС Windows. Передбачити повідомлення при неправильних діях або некоректних даних;

– у меню вікна передбачити довідку з інформацією про автора програми з фотографією автора, e-mail автора та завдання, яке виконує програма;

– розташувати вікно посередені екрана;

– використати макроси;

– використати завантаження графічних BMP-файлів з візуальним ефектом натиснення клавіші і обмеженням кількості натиснень на кнопку;

– використати статичні об'єкти діалогового вікна;

– використати SroupBox (обрамляючі рамки) у діалоговому вікні;

– переміщати вікно лівою клавішею миші;

– використати спливаюче вікно з додатковою інформацією;

– у необхідних випадках дані зберегти у файлі.

– у верхній частині створеного вікна відобразити іконку;

– у правому нижньому куті монітора (у системному tray) відобразити іконку програми, наприклад, за допомогою структури NOTIFYICONDATA;

– текст, який виводиться у вікно екрана та фон за текстом повинні бути кольоровими;

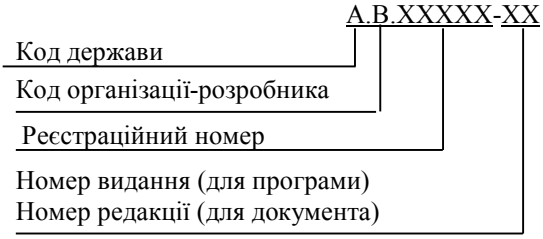
– всі рядки програми та обов'язковий алгоритм програми повинні мати коментарі;

– всі копії екранів повинні мати надписи.

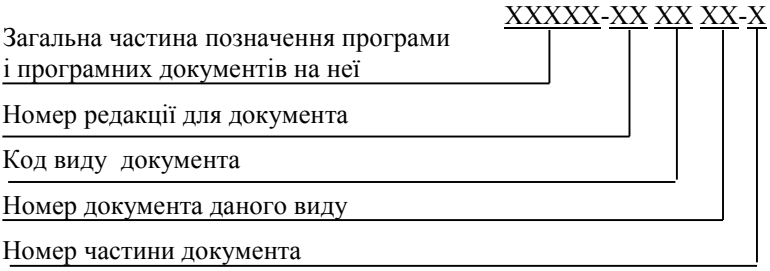
4.2. Реєстраційна система позначення програм і програмних документів

Позначення програм і документів (ДСТУ 19.103.77 ЄСПД) повинно складатися з груп знаків, розділених крапками (після коду країни та коду організації-розробника), пропусками (після номера редакції документа і коду виду документа), дефісами (після реєстраційного номера і номера документа даного виду).

Структура позначення програм і її програмного документа – ***специфікації:***



Структура позначення ***інших програмних документів:***



Номер видання програми або номер редакції документа привласнюють у порядку зростання з 01 до 99.

Код виду документа привласнюють відповідно до вимог ДСТУ 19.101-77 ЄСПД.

Номер документа даного вигляду привласнюють у порядку зростання з 01 до 99.

Номер частини одного і того ж документа привласнюють у порядку зростання з 1 до 9.

Примітка. ***Якщо документ складається з однієї частини, то дефіс і порядковий номер частини не вказують.***

5. ВИДИ ПРОГРАМ ТА ПРОГРАМНИХ ДОКУМЕНТІВ

Програму (за ДСТУ19781-90) дозволяється ідентифікувати та використати самостійно або у складі інших програм. Програми розподіляються (ДСТУ 19.101-77 ЄСПД зі змінами №1, затвердженими 06.1981) на види: компонент та комплекс.

Компонент – це програма, яка розглядається як єдине ціле, яка виконує закінчену функцію та використається самостійно або в складі комплексу.

Комплекс – це програма, яка складається з двох або більше компонентів та (або) комплексів, які виконують взаємозв'язуючі функції, та буде використана самостійно або в складі іншого комплексу.

Види програмних документів та їх зміст наведені в табл. 5.1.

Таблиця 5.1 – Види програмних документів

Вид програмного документа	Зміст програмного документа
Специфікація	Склад програми та її документація
Відомість утримувачів оригіналів	Перелік підприємств, на яких зберігаються оригінали програмних документів
Текст програми	Запис програми з необхідними коментарями
Опис програми	Відомості про логічну структуру та функціонування програми
Програма та методика випробувань	Вимоги, що підлягають перевірці при випробуванні програми, а також порядок і методи їх контролю
Технічне завдання	Призначення та область використання програми, технічні, техніко-економічні та спеціальні вимоги, які висуваються до програми, необхідні стадії та строки розробки, види випробувань
Пояснювальна записка	Схема алгоритму, загальний опис програми та (або) функціонування програми, а також обґрунтування прийнятих технічних та техніко-економічних рішень
Експлуатаційні документи	Відомості для забезпечення функціонування та експлуатації програми

Основні види програмних документів, що розробляються на різних стадіях і їх коди наведені в табл. 5.2.

Дозволяється об’єднувати деякі види експлуатаційних документів. Необхідність об’єднання цих документів указується в технічному завданні. Об’єднаному документу привласнюється найменування та позначення одного з використаних документів.

Таблиця 5.2 – Основні види програмних документів, що розробляються на різних стадіях і їх коди

Код	Вид документа	Стадії розробки			
		Ескізний проект	Технічний проект	Робочий проект	
				Компонент	Комплекс
–	Специфікація	–	–	○●	●
12	Текст програми	–	–	●	○
13	Опис програми	–	–	○	○
32	Керівництво системного	–	–	○	○
33	програміста	–	–	○	○
34	Керівництво програміста	–	–	○	○
81	Настанови оператору	○	○	–	–
90-99	Пояснювальна записка	○	○	○	
	Інші документи				

Умовні позначення:

● – документ обов’язковий;

○● – документ обов’язковий для компонентів, що мають самостійне застосування;

○ – необхідність складання документа визначається на етапі розробки і затвердження технічного завдання;

– – документ не складають.

5.1. Перелік документів

За результатами проектування програмного забезпечення (ПЗ) розробником складається програмна документація.

До складу обов’язкових програмних документів курсового проекту (КП) за навчальною дисципліною “Системне програмування” входять:

- специфікація (код документа відсутній);
- технічне завдання (код документа – 90);

- пояснювальна записка (код документа – 81);
- керівництво системного програміста (код документа – 32);
- текст програми (код документа – 12);
- схема алгоритму (код документа – від 91 по 99).

За необхідності згідно з вимогами керівника складаються інші програмні документи:

- настанови оператору;
- керівництво програміста;
- програма та методика випробувань та ін.

5.2. Вимоги до оформлення програмних документів

Склад та зміст програмних документів установлені ДСТУ 19.101-77 ЄСПД. Програмні документи оформлюються згідно з вимогами ЄСПД («Єдиної системи програмної документації»). Оформлення текстових документів виконується згідно з ДСТУ 3008–95 (http://ru. wikipedia.org/wiki/Единая_система_программной_документации).

Примітки:

1) у всіх додатках сполучення КІТ2ХХ означає групу, в якій навчається студент, наприклад КІТ27а, а сполучення ХХХХХ – номер залікової книжки;

2) відповідно до вимог відносно текстових документів вони виконуються з міжрядковим інтервалом 1,5, але у виконанні курсового проектування допускається розробка документа з інтервалом 1,0;

3) згідно з вимогами до текстових документів кожен розділ тексту має починатися з нової сторінки, але при малому обсязі розділу допускається розміщення на одній сторінці декількох розділів.

6. ПРОГРАМНІ ДОКУМЕНТИ

6.1. Склад програмних документів

Матеріали програмного документа (ГОСТ 19.105-78 ЄСПД) розташовують у такій послідовності:

- титульна частина:
  - аркуш затвердження (не входить у загальну кількість аркушів документа);
  - титульний аркуш (перший аркуш документа);
- інформаційна частина:
  - анотація українською та російською мовами. Анотація повинна бути присутня, якщо вона є обов'язковою для даного документа. В інших випадках необхідність анотації визначається керівником та студентом;
  - зміст;
- основна частина;
- частина реєстрації змін:
  - лист реєстрації змін.

Перелік умовних позначень, символів, одиниць, скорочень і термінів (необхідність даного елемента визначається керівником проекту) визначається стандартами ЄСПД на відповідні документи.

Структура основної частини програмного документа встановлюється стандартами ЄСПД на відповідні документи.

Склад додатків визначається керівником проекту та студентом.

6.2. Види програмних документів

6.2.1. Специфікація

Специфікація розробляється згідно з вимогами ГОСТ 19.202-78 ЄСПД зі змінами №1, затвердженими 09.1981.

Специфікація (див. Додаток Д) є основним програмним документом для комплексів і для компонентів, що застосовуються самостійно. Для компонентів, що не мають специфікації, основним програмним документом є «Текст програми».

Структура та оформлення специфікації встановлюється відповідно

до ГОСТ 19.105-78 ЄСПД та повинна мати:

- аркуш затвердження;
- титульний аркуш

й розділи:

- документація;
- комплекси;
- компоненти,

які заносяться до форми специфікації.

Найменування кожного розділу вказують у вигляді заголовка в графі «Найменування». Для документів, виконаних друкованим способом, заголовок підкреслюють.

У розділ «Документація» вносять програмні документи на дану програму, ***крім специфікації і технічного завдання, у порядку зростання коду виду документа***, що входить у позначення. Далі записують запозичені програмні документи. Запис їх виконується відповідно до зростання кодів організацій (підприємств) – розробників і далі відповідно до коду виду документа, який входить у позначення. Після кожного розділу специфікації необхідно залишати трохи вільних рядків для додаткових записів.

Графи специфікацій заповнюють у такий спосіб:

1) у графі «Позначення» вказують:

- у розділі «Документація» – позначення документів програми;
- у розділі «Комплекси» – позначення специфікації комплексів, що входять у даний комплекс;
- у розділі «Компоненти» – позначення основних програмних документів компонентів;

2) у графі «Найменування» вказують:

- у розділі «Документація» – найменування та вид документа для документів на дану програму; повне найменування програми, найменування і вид документа для запозичених документів;
- у розділах «Комплекси» та «Компоненти» – повне найменування програми, найменування та вид документа;

3) у графі «Примітка» – додаткові відомості, що належать до записаних у специфікації програмах.

За відсутності місця в графі «Примітка» допускається записувати тільки порядкові номери приміток. Текст приміток записують у кінці відповідних розділів специфікації. Допускається текст приміток записувати на останніх аркушах специфікації без граф із зазначенням порядкового номера примітки. У графі «Позначення» запис роблять в один рядок. В інших графах специфікації записи можна робити у

декількох рядках.

Приклад оформлення специфікації наведено в Додатку Б.

6.2.2. Технічне завдання

Технічне завдання розробляється згідно з вимогами ДСТУ 19.201-78 ЄСПД зі змінами №1, затвердженими 06.1981 (http://ru.wikipedia.org/wiki/ Единая_система_программной_документации). Номери аркушів (сторінок) проставляються ***у верхній частині аркуша над текстом***.

6.2.2.1. Вимоги до змісту й оформлення технічного завдання

Технічне завдання повинно включати:

- аркуш затвердження;
- титульний аркуш;
- анотацію (є необов'язковою);
- зміст;
- основну частину:
  - вступ;
  - підстава для розробки;
  - призначення розробки;
  - вимоги до програми або програмного виробу;
  - вимоги до програмної документації;
  - техніко-економічні показники;
  - стадії та етапи розробки;
  - порядок контролю та приймання.

6.2.2.2. Розділи основної частини технічного завдання

Основна частина технічного завдання складається з таких підрозділів:

а) вступ, де подається: найменування, коротка характеристика області застосування програми чи програмного виробу або об'єкта, в якому використовують програму чи програмний виріб;

б) підстава для розробки, де вказуються:

- документ (документи), на підставі якого ведеться розробка;
- організація, що затвердила документ і дата його затвердження;
- найменування та умовне позначення теми;

в) призначення розробки, де подається функціональне й експлуатаційне призначення програми;

г) вимоги до програми або програмного виробу включають такі підрозділи:

- вимоги до функціональних характеристик (вимоги до складу

функцій, що виконуються, організації вхідних і вихідних даних, часових характеристик і т.ін.);

- вимоги до надійності (вимоги до забезпечення надійного функціонування: забезпечення стійкого функціонування, контроль вхідної і вихідної інформації, час відновлення після відмовлення і т.п.);
- умови експлуатації (температура повітря, відносна вологість і т.ін. для обраних носіїв даних, при яких повинні забезпечуватися задані характеристики, а також вид обслуговування, необхідна кількість та кваліфікація персоналу);
- вимоги до складу та параметрів технічних засобів (необхідний склад технічних засобів із указівкою їх основних технічних характеристик);
- вимоги до інформаційної і програмної сумісності (вимоги до інформаційних структур на вході і виході та методів рішення, вихідних кодів мов програмування і програмних засобів, що використовуються програмою, за необхідності треба забезпечувати захист інформації та програм);
- вимоги до маркування й пакування (вимоги до маркування програмного виробу, варіанти і способи пакування);
- вимоги до транспортування і збереження (для програмного виробу вказуються умови транспортування, місця збереження, умови збереження, умови складування, терміни збереження в різних умовах);
- спеціальні вимоги;

д) вимоги до програмної документації, потребують указувати попередній склад програмної документації і за необхідності спеціальні вимоги до неї;

ж) техніко-економічні показники передбачають розрахунок:

- орієнтованої економічної ефективності;
- передбачуваної річної потреби;
- економічних переваг розробки в порівнянні з кращими вітчизняними і закордонними зразками й аналогами.

з) стадії й етапи розробки: вказати необхідні стадії розробки, етапи і зміст робіт (перелік програмних документів, що мають бути розроблені, узгоджені і затверджені), а також, як правило, терміни розробки і визначення виконавців;

к) порядок контролю і приймання: необхідно вказати види тестування і загальні вимоги до приймання роботи.

Примітка: деякі розділи та підрозділи технічного завдання, що не є необхідними під час курсового проектування за дозволом керівника можна до документа не включати.

Приклад оформлення технічного завдання наведено в Додатку В.

6.2.3.Пояснювальна записка

6.2.3.1. Вимоги до змісту й оформлення пояснювальної записки

Вимоги до змісту й оформлення пояснювальної записки наведені в ДСТУ 19.404-79 ЄСПД.

Структура та оформлення пояснювальної записки встановлюється відповідно до ДСТУ 19.105-78 ЄСПД та повинна мати:

- аркуш затвердження;
- титульний аркуш;
- анотацію (є необов'язковою);
- зміст;
- основну частину.

6.2.3.2. Розділи основної частини пояснювальної записки

Основна частина документа містить розділи:

а) вступ, що включає: найменування програми і (або) умовне позначення теми розробки; документи, на підставі яких ведеться розробка з указівкою організації і датою затвердження;

б) призначення та область застосування, де вказується: призначення програми; коротка характеристика області застосування програми;

в) технічні характеристики, що містять:

- постановку задачі на розробку програми, опис застосованих математичних методів і за необхідності опис допущень і обмежень, пов'язаних з обраним математичним апаратом;
- опис алгоритму і (або) функціонування програми з обґрунтуванням вибору схеми алгоритму рішення задачі, можливі взаємодії програми з іншими програмами;
- опис і обґрунтування вибору методу організації вхідних і вихідних даних;
- опис і обґрунтування вибору складу технічних і програмних засобів на підставі проведених розрахунків і (або) аналізів, розподіл носіїв даних, які використовує програма;

г) очікувані техніко-економічні показники, де зазначаються техніко-економічні показники, що зумовлюють перевагу обраного варіанта технічного рішення, а також за необхідності очікувані оперативні показники;

д) джерела, використані під час розробки – перелік науково-технічних публікацій, нормативних документів та інших науково-технічних матеріалів, на які є посилання в основному тексті;

е) додатки, в яких розміщують таблиці, обґрунтування, методики,

розрахунки й інші документи, використані під час розробки.

За необхідності до складу документа включаються схеми алгоритмів, що виконуються згідно з вимогами ЄСПД ДСТУ 19.701-90 ЄСПД (http://ru.wikipedia.org/wiki/Единая_система_программной_документации).

Зразок оформлення пояснювальної записки наведено в Додатку Г.

6.2.4. Опис програми

6.2.4.1. Вимоги до змісту й оформлення опису програми

Опис програми виконується відповідно до ДСТУ 19.402-78 ЄСПД. Структура та оформлення документа встановлюється відповідно до ГОСТ 19.105-78 ЄСПД.

Опис програми містить:

- аркуш затвердження;
- титульний аркуш;
- анотацію;
- зміст;
- основну частину.

6.2.4.2. Розділи основної частини опису програми

Опис програми містить такі розділи:

а) загальні відомості, де вказуються: позначення і найменування програми; програмне забезпечення, необхідне для функціонування програми; мови програмування, якими написана програма;

б) функціональне призначення, де відображається: класи розв'язуваних задач і призначення програми; відомості про функціональні обмеження на застосування;

в) опис логічної структури, що містить: алгоритм програми; методи, що використовуються; структуру програми з описом функцій складових частин і зв'язку між ними; зв'язок програми з іншими програмами;

г) технічні засоби, що використовуються: типи електронних обчислювальних машин і пристроїв, що використовуються під час розробки програми;

д) виклик і завантаження включає: спосіб виклику програми з відповідного носія даних; точки входу в програму; адреси завантаження; інформацію про використання оперативної пам'яті; обсяг програми;

ж) вхідні дані: характер, організація і попередня підготовка вхідних даних; формат, опис і спосіб кодування вхідних даних;

з) вихідні дані: характер і організація вихідних даних; формат, опис і спосіб кодування вихідних даних.

Допускається вміст розділів ілюструвати пояснювальними прикладами, таблицями, схемами, графіками.

6.2.5. Текст програми

Текст програми розробляється відповідно до ГОСТ 19.401-78 ЄСПД. Структура та оформлення документа встановлюється відповідно до ГОСТ 19.105-78 ЄСПД.

Текст програми включає:

- аркуш затвердження;
- титульний аркуш;
- анотацію;
- зміст;
- основну частину.

Основна частина документа має складатися з текстів одного чи декількох розділів, яким даються назви.

Кожен з цих розділів реалізується одним із типів символічного запису, наприклад:

– символічний запис вихідною мовою;

– символічний запис проміжними мовами;

– символічне подання машинних кодів і т.ін.

У символічний запис розділів рекомендується включати коментарі, що полегшують сприйняття, наприклад, функціонального призначення, структури програми або окремих її частин.

У документі наводити тільки ті частини програми, які розроблені безпосередньо студентом.

Приклад оформлення тексту програми наведено в Додатку Д.

6.2.6. Настанови оператору

Настанови оператору розробляються згідно з вимогами ГОСТ 19.505-79 ЄСПД (http://ru.wikipedia.org/wiki/Единая_система_программной_документации). Структура та оформлення документа встановлюється відповідно до ГОСТ 19.105-78 ЄСПД.

Настанови оператору складаються з:

- аркуша затвердження;
- титульного аркуша;
- анотації;
- змісту;
- основної частини.

Основна частина складається з таких розділів:

а) призначення програми, де наводяться відомості про призначення

програми й інформація, достатня для розуміння функцій програми і її експлуатації;

б) умови виконання програми, а саме: умови, необхідні для виконання програми (мінімальний і (або) максимальний склад апаратних і програмних засобів тощо);

в) виконання програми, тобто: послідовність дій оператора, що забезпечують завантаження, запуск, виконання і завершення програми; опис функцій, формату і можливих варіантів команд, за допомогою яких оператор здійснює завантаження і керує виконанням програми, а також відповідями програми на ці команди;

г) повідомлення оператору, а саме: тексти повідомлень, що з'являються у ході виконання програми, опис їх змісту та відповідні дії оператора (дії оператора у випадку збою, можливості повторного запуску програми і т.ін.).

Допускається зміст розділів ілюструвати прикладами, що пояснюють викладені тези, таблицями, схемами, графіками.

У додатки допускається включати різні матеріали, що недоцільно включати в розділи керівництва оператору.

6.2.7. Керівництво системного програміста

Керівництво системного програміста розробляється згідно з вимогами ДСТУ 19.503-79 ЄСПД. Структуру і оформлення документа встановлюють відповідно до ДСТУ 19.105-78 ЄСПД (http://kladovka.net.ru/index.cgi?pid= list&rid=61).

Складання інформаційної частини (анотації і зміст) є обов'язковим.

Керівництво системного програміста повинно містити такі розділи:

- загальні відомості про програму;
- структура програми;
- настройка програми;
- перевірка програми;
- додаткові можливості;
- повідомлення системного програміста.

Залежно від особливостей документів допускається об'єднувати окремі розділи або вводити нові.

В обґрунтованих випадках допускається розділ «Додаткові можливості» не приводити, а в найменуваннях розділів випускати слово «програма» або замінювати його на «найменування програми».

У розділі «Загальні відомості про програму» повинно бути вказано призначення і функції програми й зведення про технічні і програмні засоби, що забезпечують виконання даної програми.

У розділі «Структура програми» повинні бути наведені зведення про структуру програми, її складових частинах, про зв'язки між складовими частинами і про зв'язки з іншими програмами.

У розділі «Настройка програми» повинно бути наведено опис дій з настроювання програми на умови конкретного застосування (настроювання на склад технічних засобів, вибір функцій та ін.).

За необхідністю наводять пояснюючі приклади.

У розділі «Перевірка програми» повинно бути наведено опис способів перевірки, що дозволяє дати загальний висновок про працездатність програми (контрольні приклади, методи прогону, результати).

У розділі «Додаткові можливості» повинно бути наведено опис додаткових розділів функціональних можливостей програми та способів їх вибору.

У розділі «Повідомлення системного програміста» повинні бути вказані тексти повідомлень, що видаються в ході виконання настройки, перевірки програми, а також у ході виконання програми, опис їх змісту і дій, які необхідно зробити за цими повідомленнями.

У додатку до керівництва системного програміста можуть бути наведені додаткові матеріали (приклади, ілюстрації, таблиці, графіки і т.п.).

7. СХЕМИ АЛГОРИТМІВ І ПРОГРАМ

Правила виконання схем алгоритмів і програм, систем програмного забезпечення обчислювальних машин, комплексів і систем незалежно від їх призначення та області застосування встановлюють стандарти системи програмного забезпечення (СПЗ). При виготовленні цих схем окремі функції алгоритмів і програм зображуються у вигляді умовних позначень – символів.

Співвідношення геометричних розмірів символів (висоти "*a*" і ширини "*b*") мають бути чітко витримані відповідно до вимог СПЗ. Розмір "*a*" має вибиратися із ряду 10, 15, 20 мм. Допускається збільшувати розмір "*a*" на число кратне 5. Розмір "*b*" дорівнює 1,5*a*. У межах однієї схеми, при виконанні її від руки, допускається застосовувати не більше ніж два суміжних розміри ряди чисел, кратних до 5.

Для полегшення накреслення і знаходження символів на схемі рекомендується поле аркуша розбивати на зони. Розмір зон установлюється з урахуванням мінімальних розмірів символів, зображених на даному аркуші. Координати зон, які позначають ряди та колонки у вигляді сполучення букв і цифр (А01, В02 та ін.), присвоюються символам і вписуються в розрив контуру символу вверху зліва. Якщо поле аркуша схеми не розбито на зони, то символам привласнюються порядкові номери. Допускається один символ розміщати в двох і більше зонах, якщо розмір символу перевищує розмір зони.

Символ рекомендується розміщувати в середині зони за можливості симетрично і в такому положенні, як він зображений у стандарті. Виняток складають символи "Лінії потоку", "Канал зв'язку", "Коментар", "Міжсторінковий сполучник", "Транспортування носіїв", "Матеріальний потік".

Лінії потоку мають бути паралельними до ліній рамки формату. Напрямки ліній потоку зверху вниз і зліва направо приймають як основні і, якщо вони не мають зламів, стрілками можна не позначати. В інших випадках напрямок ліній потоку позначати стрілкою обов'язково. Відстань між паралельними лініями має бути не менше ніж 3 мм, між рештою символами схеми – не менше ніж 5 мм. Місце злиття ліній потоку позначається точкою.

У середині символу поміщають символічний або словесний запис алгоритму. Якщо запис, який відповідає символу, не вміщається в середині символу або необхідні певні пояснення, то біля символу (справа

або зліва) на вільному полі схеми пишуться коментарі (символ “Коментар”). Коментарі записують паралельно з основним записом. Коментарі можуть належати і до ліній потоку.

Біля символу “Рішення” над кожною вхідною лінією потоку або справа від неї проставляють можливі результати (ознака умови рішення), наприклад: Так, Ні, =, > ,< . За наявності кількості можливих результатів більше ніж три умови проставляються в розриві ліній потоку.

Записи з середини символу і поруч з ним мають бути короткими і виконані креслярським шрифтом. Скорочення слів і абревіатури, за винятком установлених державними стандартами, мають бути розшифровані в нижній частині поля схеми або в документі, до якого ця схема належить.

Якщо схеми перенасичені символами, лінії потоку допускається обривати. При цьому в кінці (на початку) обриву поміщається символ “З’єднувач” із позначенням у вигляді букви і цифри, які вказують координату зони символу, до якого (або від якого) йде лінія потоку.

Якщо схеми зміщені на кількох аркушах, обриви ліній потоку супроводжуються символом “Міжсторінковий з’єднувач”, всередині якого в першому рядку записується номер аркуша, а в другому рядку – координати зони символу, до якого або від якого йде лінія потоку.

У схемі символу може бути ідентифікатор, який має поміщатися зліва над символом (наприклад, для посилання в інших частинах документації) (рисунок 7.1).

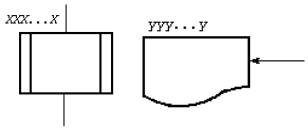


Рисунок 7.1 – Приклад для посилання в інших частинах документації

8. ТЕМАТИКА ПРОЕКТУВАННЯ

В інтересах кафедри та організацій, що працюють з нею, допускається виконання проекту за ініціативною чи замовленою темою. Тема і зміст курсового проекту в цьому випадку узгоджується з лектором як відповідальним за навчальну дисципліну. Теми курсових проектів мають відповідати актуальним проблемам системного проектування. Формулювання теми має бути коротким, конкретним і відображати суть професійних задач, що вирішуються. Пункти меню програми необхідно узгодити з керівником проекту.

Основні напрямки тем курсових проектів можна поділити на такі типи:

Тип 1

1.1. Програма keylogger, яка запам'ятовує всі натискання на кнопки мишки з назвою імен вікон та часу, коли ця дія виконана.

1.2. Програма keylogger, яка відправляє текст з комп'ютера на визначену почтову скриню.

1.3. Програма з виведенням інформації про пам'ять комп'ютера.

1.4. Програма з виведенням інформації про дисководи.

1.5. Програма керування відеоадаптером.

1.6. Програма виведення сигналів азбуки Морзе на системний динамік.

1.7. Програма керування мікшером звукової плати.

1.8. Програма керування послідовним портом.

1.9. Пошук файлів на диску та їх перейменування.

1.10. Програма для відображення системних номерів (хендлів) об'єктів, іконок, курсорів та ін.

1.11. Програма очищення дисків від тимчасових файлів.

1.12. Програма керування корзиною (http://www.delphimaster.ru/articles/shfileopst/).

1.13. Програма блокування Windows (http://www.proxy-base.org/f29/).

1.14. Splash-вікно при завантаженні програми (http://www.manhunter.ru).

1.15. Обробка підключення і відключення знімного накопичувача.

1.16. Програма створення та виведення феєрверків.

1.17. Отримання списку ікон в треї (http://www.manhunter.ru).

1.18. Відтворення музики в програмах (http://www.manhunter.ru).

1.19. Побудова карти пам'яті процесу (http://www.manhunter.ru).

Тип 2

При виведенні графічних об'єктів використати лінії рівнів та їх фарбування: чим вище рівень – тим колір світліше, чим нижче – темніше.

2.1. Розробити текстовий редактор, який виконує дії, аналогічні функціям блокнота від "Офису 2003", які розташовані під кнопками "Файл", "Правка", "Справка".

2.2. Розробити текстовий редактор, який виконує дії, аналогічні функціям блокнота від "Офису 2003", які розташовані під кнопками "Формат", "Вид", "Справка".

2.3. Розробити графічний редактор для растрових зображень, який виконує функції:

– читання, запис та відображення bmp-файлів;

– масштабування зображення;

– повернення зображення на заданий кут.

2.4. Розробити калькулятор з функціями, які необхідно узгодити з викладачем.

2.5. Розробити програму-провідник для роботи з файлами з виконанням функцій копіювання, перейменування та отримання інформації від файлів.

2.6. Написати програму з використанням запису та читання часу створення, останньої модифікації і останнього доступу до файлу.

2.7. Написати програму симетричного шифрування (для шифрування та дешифрування використовується один ключ) тексту повідомлення. Вивести на екран призначення програми, необхідні запити на введення та отримання результату операції.

2.8. Написати програму шифрування тексту повідомлення методом відкритого ключа (між відкритим та закритим ключами використовується математична залежність). Вивести на екран призначення програми, необхідні запити на введення та отримання результату операції.

2.9. Написати програму конвертації чисел з однієї позиційної системи числення в іншу. Основи систем числення, які можуть бути використані, такі: 2, 3, 4, 5, 6, 7, 8, 9, 10. Число для конвертації та система числення результату задаються користувачем.

2.10. Написати програму, яка дозволяє вводити текстове повідомлення, міняти напрям його руху на екрані монітора (зверху вниз, знизу вверх, зліва на право, справа наліво), змінювати товщину шрифту. Букви текстового повідомлення мають переливатися всіма кольорами.

2.11. Написати програму "Часи" (цифрові або аналогові).

2.12. Написати програму отримання різних ступенів квадратної матриці розміром 4 × 4 для цілих чисел. Видати: призначення програми, необхідні запити на введення та отримання результату операції.

2.13. Написати програму отримання різних ступенів квадратної матриці розміром 4 × 4 для дійсних чисел. Видати: призначення програми, необхідні запити на введення та отримання результату операції.

2.14. Написати програму перемноження двовимірних квадратних матриць розміром від 4 × 4 до 16 × 16 для цілих чисел. Видати: призначення програми, необхідні запити на введення та отримання результату операції.

2.15. Написати програму перемноження двовимірних квадратних матриць розміром від 4 × 4 до 16 × 16 для дійсних чисел. Видати: призначення програми, необхідні запити на введення та отримання результату операції.

2.16. Написати програму перевірки знань з шифруванням масиву питань, відповідей та оцінок із випадковою вибіркою та обмеженням часу відповіді.

2.17. Написати програму виведення тексту, над яким виконуються дії:

– колір символів літер зліва направо змінюється жовтим кольором;

– колір символів літер справа наліво змінюється червоним кольором.

2.18. Написати програму виведення кольорового тексту, який з'являється рядками та зсувається з центра екрана до верху, а його дзеркальне відображення – з центра до низу.

2.19. Написати програму, в якій відображається ефект букв, що зсипаються.

2.20. Написати програму створення адресної книги.

2.21. Написати програму "Календар".

2.22. Написати гру "Хрестики – нуліки" на полі 3 × 3.

2.23. Написати гру "Тетріс".

2.24. Написати гру "Кросворд". Якщо слово співпало, то воно відображається одним кольором, а якщо ні – червоним.

2.25. Написати гру "Змія". Змія рухається по полю та збирає об'єкти. Якщо об'єкт підібрано, то змія збільшується.

2.26. Написати програму керування холодильним пристроєм. На екрані відобразити градусник, в іншому віконці – температуру. Температура повільно зростає та при досягненні введеної величини – зменшується.

2.27. Написати програму керування світлофорами перехрестя з відображенням малюнків машин та їх переміщенням.

2.28. Написати програму керування ліфтом із зображенням кнопок керування напрямком руху, керування мотором та зображенням ліфта, який динамічно рухається.

2.29. Програма з різними градієнтними заливками вікна.

Тип 3

Особисті вимоги до програм типу 4: при рисуванні фігур використати плавну зміну кольору фігури.

3.1. Написати програму виведення у вікно 11-кінцеву кольорову зірку із зафарбованими однаковими частинами в цій фігурі.

3.2. Написати програму виведення у вікно 13-кінцеву кольорову зірку, в якій внутрішня частина плавно переливається всіма кольорами.

3.3. Написати програму виведення вікна з відображенням трикутника, залитого градієнтно (плавно) змінними кольорами: у кутах трикутника кольори – *Red, Blue, Green*, які в центрі переходять у *White* колір.

3.4. Написати програму з відображенням куба, колір площин якого динамічно змінюється.

3.5. Написати програму з відображенням кулі, колір та тіні якої динамічно змінюються.

3.6. Нарисувати зображення “Павлин”, яке складається з відрізків, при якому початкова координата відрізка X_1 змінюється в циклі від 1 до 319, а координата Y_1 – не змінюється. Друга координата обчислюється за формулою

$X_2 = 120 + 100\sin(X_1/30)$; $Y_2 = 90 + 100\cos(X_1/30)$.

Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури.

3.7. Нарисувати зображення “Мережива”. Для цього на екрані будуються вершини правильного 18-кутника, центр якого збігається з центром екрана. Кожна з вершин з’єднується відрізками зі всіма іншими вершинами. Координати вершин задаються за формулами

$X_i = X_c + R\cos(2\pi\ i/n)$;

$Y_i = Y_c + R\sin(2\pi\ i/n)$; $i = 1 \div 18$,

де i – номер вершини, R – радіус окружності, описаної біля багатокутника; X_c, Y_c – координати центра.

Передбачити статичний та динамічний режими, масштаб відображення, фон екрану та колір фігури.

3.8. Нарисувати зображення, яке складається із вкладених один в одного квадратів, що динамічно повертаються.

3.9. Нарисувати згідно з формулами спіраль Архімеда (http://ru.wikipedia.org, http://graphinpas.narod.ru) з різними параметрами.

Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури.

3.10. Нарисувати циклоїдальну криву у вигляді равлика Паскаля (http://physics.nad.ru/curves.html, http://imcs.dvgu.ru/struc/kmf/download/angeom/angeom.html) за параметричними рівняннями:

$x = (R + mR)\cos(mt) - h\cos(t + mt)$;

$y = (R + mR)\sin(mt) - h\sin(t + mt)$, при $R = 1$; $r = 1$; $h = 1.5$,

де R – радіус нерухомої окружності; r – радіус окружності, що котиться; h – відстань від центру окружності, що котиться, до крапки; $m = r/R$.

Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.11. Нарисувати циклоїдальну криву у вигляді рози (http://physics.nad.ru/curves.html) за параметричними рівняннями:

$x = (R + mR)\cos(mt) - h\cos(t + mt)$;

$y = (R + mR)\sin(mt) - h\sin(t + mt)$, при $R = 1$; $r = 1/6$; $h = 1 + 1/6$,

де R – радіус нерухомої окружності; r – радіус окружності, що котиться; h – відстань від центру окружності, що котиться, до крапки; $m = r/R$.

Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.12. Нарисувати циклоїдальну криву у вигляді епіциклоїди (http://physics.nad.ru/curves.html, http://imcs.dvgu.ru/struc/kmf/download/angeom/ angeom.html) за параметричними рівняннями:

$x = (R + mR)\cos(mt) - m\cos(t + mt)$;

$y = (R + mR)\sin(mt) - m\sin(t + mt)$, при $m = 1/10, 1/3, 2/3$,

де R – радіус нерухомої окружності; r – радіус окружності, що котиться; $m = r/R$.

Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.13. Нарисувати циклоїдальну криву у вигляді гіпоциклоїди (http://physics.nad.ru/curves.html, http://imcs.dvgu.ru/struc/kmf/download/angeom/ angeom.html) за параметричними рівняннями:

$x = (R - mR)\cos(mt) + m\cos(t - mt)$;

$y = (R - mR)\sin(mt) - m\sin(t - mt)$, при $m = 1/4, 1/2.1$,

де R – радіус нерухомої окружності; r – радіус окружності, що котиться; $m = r/R$.

Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.14. Нарисувати циклоїдальну криву у вигляді гіпоциклоїди (http://physics.nad.ru/curves.html, http://www.matematike.net/vysshaya-matema-tika/analiticheskaya-geometriya/spetsialnie-plokostnie-krivie.html) за параметричними рівняннями:

$x = R\cos^3\Theta$;

$y = R\sin^3\Theta$,

де R – радіус окружності.

Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.15. Нарисувати згідно з формулами кардіоїду (http://ru.wikipedia. циклorg/wiki/Категория:Алгебраические_кривые, http://ru. wikipedia. org/ wiki/Кардиоида). Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами). Координати вершин в параметричних координатам задаються за формулами

$X = 2R\cos t(1+\cos t)$;

$Y = 2R\sin t(1+\cos t)$,

де R – радіус окружності.

3.16. Нарисувати згідно з формулами трилисник (http://ru.wikipedia. org, http://graphinpas.narod.ru). Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури.

3.17. Нарисувати згідно з формулами чотирилисник (http://ru. wikipedia.org, http://graphinpas.narod.ru). Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури.

3.18. Нарисувати згідно з формулами фігури Ліссажу (http://ru.wikipedia.org, http://graphinpas.narod.ru). Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.19. Нарисувати згідно з формулами фігуру конхоиду (http://ru.wikipedia.org, http://graphinpas.narod.ru). Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.20. Нарисувати згідно з формулами фігуру епіциклóіди (http://ru.wikipedia.org).

Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.21. Нарисувати згідно з формулами фігуру гіпоциклоїди (http://ru.wikipedia.org, http://dic.academic.ru/dic.nsf/ruwiki) з $k = R/r = 3$; 4; 5; 6; 2,1; 3,8; 5,5; 7,2. Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.22. Нарисувати згідно з формулами фігуру астроїди (http://ru.wikipedia.org, http://dic.academic.ru/dic.nsf/ruwiki) з різними параметрами. Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.23. Нарисувати згідно з формулами фігуру Лемніската Бернуллі (http://ru.wikipedia.org, http://dic.academic.ru/dic.nsf/ruwiki) з різними параметрами. Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.24. Нарисувати згідно з формулами фігуру Лемніската Бута (http://ru.wikipedia.org, http://dic.academic.ru/dic.nsf/ruwiki) з різними параметрами. Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.25. Нарисувати згідно з формулами фігуру Лемніската Бернуллі (http://ru.wikipedia.org, http://dic.academic.ru/dic.nsf/ruwiki) з різними параметрами. Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.26. Нарисувати згідно з формулами фігуру Лемніската Жероно (http://ru.wikipedia.org, http://dic.academic.ru/dic.nsf/ruwiki) з різними параметрами. Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.27. Нарисувати згідно з формулами фігуру овалу Кассіні (http://ru.wikipedia.org, http://dic.academic.ru/dic.nsf/ruwiki) з різними параметрами. Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.28. Нарисувати згідно з формулами фігуру логарифмічної спіралі (http://ru.wikipedia.org, http://dic.academic.ru/dic.nsf/ruwiki) з різними параметрами. Передбачити статичний та динамічний режими, масштаб відображення, фон екрана та колір фігури (фігур з різними параметрами).

3.29. Написати програму завантаження зображення та наступного його перевертання.

Тип 4

Особисті вимоги до програм типу 4: підписати номери стовпців матриць та підрахувати й вивести кількість одиниць у рядках, в стовпцях й загальну суму одиниць у всіх рядках.

4.1. Написати програму генерування поліномів за модулем два з максимальним ступенем 6 (degP(x) = 6) та матриць їх станів.

4.2. Написати програму генерування поліномів за модулем два з максимальним ступенем 7 (degP(x) = 7) та матриць їх станів.

4.2. Написати програму генерування поліномів за модулем два з максимальним ступенем 8 (degP(x) = 8) та матриць їх станів.

4.3. Написати програму генерування поліномів за модулем два з максимальним ступенем 9 (degP(x) = 9) та матриць їх станів.

4.4. Написати програму генерування поліномів за модулем два з максимальним ступенем 10 (degP(x) = 10) та матриць їх станів.

4.5. Написати універсальну програму генерування поліномів за модулем два з максимальним ступенем *n* та матриць їх станів.

4.6. Розробка генератора псевдовипадкових послідовностей, що використовує розкладання чисел алгебри.

4.7. Розробка генератора псевдовипадкових послідовностей, що використовує розкладання трансцендентних чисел.

4.8. Реалізація методу Шуфа-Аткина-Елкиса обчислення порядку групи точок еліптичної кривої.

Тип 5

Особисті вимоги до програм типу 3: підписати номери стовпців матриць та підрахувати й вивести кількість одиниць та двійок у рядках, в стовпцях й загальну суму одиниць та двійок у всіх рядках.

Написати програму генерування псевдовипадкової послідовності за правилом утворюючого полінома та додаванням за модулем три:

5.1. $x^4 \oplus_3 x \oplus_3 1$;
5.2. $x^4 \oplus_3 x^3 \oplus_3 1$;
5.3. $2x^4 \oplus_3 2x^3 \oplus_3 1$;
5.4. $2x^4 \oplus_3 2x \oplus_3 1$;
5.5. $x^4 \oplus_3 x^3 \oplus_3 2x^2 \oplus_3 2x \oplus_3 1$;
5.6. $x^4 \oplus_3 2x^3 \oplus_3 x^2 \oplus_3 2x \oplus_3 1$;
5.7. $2x^4 \oplus_3 x^3 \oplus_3 2x^2 \oplus_3 x \oplus_3 1$;
5.8. $2x^4 \oplus_3 2x^3 \oplus_3 x^2 \oplus_3 x \oplus_3 1$;
5.9. $2x^5 \oplus_3 x \oplus_3 1$;
5.10. $2x^5 \oplus_3 x^4 \oplus_3 1$;
5.11. $x^5 \oplus_3 2x^4 \oplus_3 x \oplus_3 1$;
5.12. $x^5 \oplus_3 x^4 \oplus_3 2x^2 \oplus_3 1$;
5.13. $2x^5 \oplus_3 x^4 \oplus_3 x^2 \oplus_3 x \oplus_3 1$;
5.14. $x^5 \oplus_3 2x^4 \oplus_3 x^3 \oplus_3 x^2 \oplus_3 1$;
5.15. $2x^5 \oplus_3 2x^4 \oplus_3 x^3 \oplus_3 x^2 \oplus_3 2x \oplus_3 1$;
5.16. $x^5 \oplus_3 2x^4 \oplus_3 2x^3 \oplus_3 x^2 \oplus_3 2x \oplus_3 1$.

5.17. Написати програму генерування поліномів за модулем три з максимальним ступенем 5 та матриць їх станів.

5.18. Написати програму генерування поліномів за модулем три з максимальним ступенем 6 та матриць їх станів.

5.19. Написати програму генерування поліномів за модулем три з максимальним ступенем 7 та матриць їх станів.

5.20. Написати програму генерування поліномів за модулем три з максимальним ступенем 8 та матриць їх станів.

5.21. Написати універсальну програму генерування поліномів за модулем три з максимальним ступенем *n* та матриць їх станів.

Додаток А

Зразок оформлення комплекту документації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРЇНИ
Національний технічний університет
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра обчислювальної техніки та програмування

КУРСОВИЙ ПРОЕКТ З КУРСУ
"СИСТЕМНЕ ПРОГРАМУВАННЯ"
НА ТЕМУ:
ПРОГРАМА ВИЗНАЧЕННЯ МАТРИЦЬ СТАНІВ ПОЛІНОМІВ
4-ГО СТЕПЕНЯ В ПОЛІ GF(3)

(комплект документації)

Розробники

Керівник проекту
доц. каф. ОТП, к.т.н., доц. Рисований О.М.

Виконавець
студент групи КІТ00А Іванов І. І.
№ залікової книжки 077777

Харків 2013

Додаток Б

Зразок оформлення специфікації

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРЇНИ
Національний технічний університет
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра обчислювальної техніки та програмування

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТП
_____ / Домнін Ф.А. /
"____"_____2013 р.

ПРОГРАМА ВИЗНАЧЕННЯ МАТРИЦЬ СТАНІВ ПОЛІНОМІВ
4-ГО СТЕПЕНЯ В ПОЛІ GF(3)

СПЕЦИФІКАЦІЯ
АРКУШ ЗАТВЕРДЖЕННЯ
КІТ00А.077777-01

Розробники

Керівник проекту
доц. каф. ОТП
_____ к.т.н., доц. Рисований О.М.

Виконавець
студент групи КІТ00А
_____ Іванов І. І.
“____” _____ 2013 р.

Харків 2013

ЗАТВЕРДЖЕНО
КІТ00А.077777-01

ПРОГРАМА ВИЗНАЧЕННЯ МАТРИЦЬ СТАНІВ ПОЛІНОМІВ 4-ГО СТЕПЕНЯ В ПОЛІ GF(3)

СПЕЦИФІКАЦІЯ
КІТ00А.077777-01
Аркушів 2

Харків 2013

Позначення	Найменування	Примітка
	Документація	
КІТ00А.077777-01 90 01	Технічне завдання	
КІТ00А.077777-01 81 01	Пояснювальна записка	
КІТ00А.077777-01 12 01	Текст програми	

Додаток В

Приклад оформлення технічного завдання

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРЇНИ
Національний технічний університет
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра обчислювальної техніки та програмування

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТП

_____ / Домнін Ф.А. /
"____"_____2013 р.

ПРОГРАМА ВИЗНАЧЕННЯ МАТРИЦЬ СТАНІВ ПОЛІНОМІВ
4-ГО СТЕПЕНЯ В ПОЛІ GF(3)
ТЕХНІЧНЕ ЗАВДАННЯ

АРКУШ ЗАТВЕРДЖЕННЯ
КІТ00А.077777-01 90 01

Розробники

Керівник проекту
доц. каф. ОТП
_____ к.т.н., доц. Рисований О.М.

Виконавець
студент групи КІТ00А
_____ Іванов І. І.
“____” _____ 2013 р.

Харків 2013

ЗАТВЕРДЖЕНО
КІТ00А.077777-01 90 01

ПРОГРАМА ВИЗНАЧЕННЯ МАТРИЦЬ СТАНІВ ПОЛІНОМІВ 4-ГО СТЕПЕНЯ В ПОЛІ GF(3)

ТЕХНІЧНЕ ЗАВДАННЯ

КІТ00А.077777-01 90 01
Аркушів 5

Харків 2013

ЗМІСТ

ВСТУП

Галузь застосування програмного продукту – тестова діагностика, захист цифрової інформації. Програмний продукт дозволяє генерувати гамміруючі послідовності при перетворенні інформації за схемою, найбільш близькій до схеми ”абсолютно стійкого” шифру в полі GF(3).

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є індивідуальне завдання на курсове проектування, видане керівником.

Розробка ведеться відповідно до графіка навчального плану на 2012 р. кафедри ОТП курсу “Системне програмування”.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення програмного продукту – генерації послідовностей, формування ключової інформації, на секретності і якості якої грунтується стійкість кріптоалгорітмов.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Програма повинна мати меню програми з чотирма основними пунктами та видавати відповідні повідомлення. Програма повинна складатися з двох частин: основної частини програми та файлу ресурсів цієї програми. Мови програмування програми – макроасемблер masm32 та C++. Середовища програмування – masm32Editor й Visual Studio 2010.

Вимоги до основної частини програми:

1. Розрахувати поліноми і їх матриці станів degP(x) = 4 в GF(3);
2. Розрахувати довжину циклу поліномів;
3. Вивести нестандартне вікно;
4. Вивести призначення програми в нестандартне вікно;
5. Використовувати 4 основних кнопки меню: “Автор”, “Расчеты”, “Справка”, “Delete”;
6. Використати дві додаткові кнопки “Расчеты”, “Delete”.

Вимоги до інтерфейсу вікна:

1. Передбачити можливість переміщення нестандартного вікна;
2. Розміщення нестандартного вікна на середині екрана;
3. Використовувати власну іконку;
4. Використовувати курсор у вигляді хреста;
5. Переміщати вікно лівою клавішею миші;
6. Використовувати перевернене спрощене вікно.

3.2 Вимоги до надійності

Програмний продукт має надійно функціонувати в програмно-апаратному середовищі, бути стійким до збоїв і не мати можливості руйнувати роботу операційної системи.

3.3 Умови експлуатації

Програмний продукт повинен функціонувати в нормальних умовах для персоналу:

– температура навколишнього середовища від 18^{O} С до 28^{O} С;

– вібрації, зовнішні магнітні, радіаційні та електричні поля не повинні перевищувати норми.

Для нормальної експлуатації програмного продукту обслуговуючому персоналу необхідні знання з експлуатації ПК.

3.4 Вимоги до складу і параметрів технічних засобів

Для експлуатації програмного продукту необхідна ЕОМ з мінімальними апаратними характеристиками:

– стандартна конфігурація IBM-сумісних ПК;

– процесор з частотою не менше 750 МГц;

– ОЗП обсягом не менше 64 Мбайт;

– жорсткий диск обсягом не менше 2 Гбайт;

– сумісні монітор та відеоадаптер;

– маніпулятор типу “миша”.

3.5 Вимоги до інформаційної та програмної сумісності

Програмний продукт, що розробляється, повинен функціонувати на ПЕОМ типу IBM PC з операційною системою Microsoft Windows 7 та більш пізніх версій у середовищі розробки MS Visual Studio 2012.

3.6 Вимоги до маркування та упаковки

Вимоги до маркування та упаковки програмного продукту ″Програма визначення матриць станів поліномів 4-го степеня в полі GF(3)″ не висуваються.

3.7 Вимоги до транспортування та збереження

Вимоги до транспортування та збереження програмного продукту ″Програма розрахунку станів поліномів 4-го степеня″ не висуваються.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація програмного продукту повинна містити такі документи:

– специфікація;

– опис програми;

– керівництво оператора;

– текст програми.

Додаткові вимоги до програмної документації.

Розробити Звіт про виконання курсового проекту. У документі Звіт виклад основних розділів повинен займати не менше 25 сторінок (без урахування додатків).

5 ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Техніко-економічні показники повинні бути визначені в процесі розробки і зазначені у відповідному розділі звіту про виконання дипломного проекту.

6 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

При прийманні роботи перевіряється: комплектність документації, відповідність вимогам, які висунуті попередньо.

Додаток Г

Зразок оформлення пояснювальної записки

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРЇНИ
Національний технічний університет
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра обчислювальної техніки та програмування

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТП
_____ / Домнін Ф. А. /
“____” _____ 2013 р.

ПРОГРАМА ВИЗНАЧЕННЯ МАТРИЦЬ СТАНІВ ПОЛІНОМІВ
4-ГО СТЕПЕНЯ В ПОЛІ GF(3)

ПОЯСНЮВАЛЬНА ЗАПИСКА

АРКУШ ЗАТВЕРДЖЕННЯ
КІТ00А.077777-01 81 01

Розробники

Керівник проекту
доц. каф. ОТП
_____ к.т.н., доц. Рисований О. М.

Виконавець
студент групи КІТ00А
_____ Іванов І. І.
“____” _____ 2013 р.

Харків 2013

ЗАТВЕРДЖЕНО
КІТ00А.077777-01 81 01

ПРОГРАМА ВИЗНАЧЕННЯ МАТРИЦЬ СТАНІВ ПОЛІНОМІВ 4-ГО СТЕПЕНЯ В ПОЛІ GF(3)

ПОЯСНЮВАЛЬНА ЗАПИСКА

КІТ00А.077777-01 81 01
Аркушів 9

Харків 2013

АНОТАЦІЯ

Розроблено програмний продукт, який дозволяє прорахувати стани всіх поліномів 4-го степеня в кінцевому полі GF(3) для подальшої розробки нелінійного паралельного сигнатурного аналізатора. Розробку виконано на мові асемблера з використанням файлу ресурсів на мові С++ та середовищ розробки masm32 та Visual Studio 2010. Пояснювальна записка містить обґрунтування методу реалізації поставленої задачі, опис роботи програми, її основних функцій, а також список використаних джерел інформації.

THE ANNOTATION

A software product which allows to count consisting of all of polynomials of 4th degree of the field of GF(3) for subsequent development parallel signatur an analyzer is developed. Development is executed in assembly language with the use of file of resources in language of S++ and environments of development of masm32 and Visual Studio 2010. An explanatory message contains the ground of metoda realization of the put task, description of work of the program, it basic functions, and also list of the utillized information generators.

ЗМІСТ

ВСТУП

Програмний продукт створено згідно з завданням на курсове проектування з дисципліни “Системне програмування”.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Програма призначена для розрахунку поліномів, їх матриць станів та довжин циклів генерування в полі Галуа GF(3). Отримані поліноми можна використати у:

– генераторах псевдовипадкових послідовностей для тестового діагностування;

– при генерації послідовностей для перетворення інформації за схемою, найбільш близькій до схеми ”абсолютно стійкого” шифру;

– хешуванні інформації;

– формуванні ключової інформації, на секретності і якості якої ґрунтується стійкість криптоалгоритмів;

– формуванні випадкових запитів при реалізації великої кількості криптографічних протоколів;

– внесенні невизначеності й т.ін.

2 ТЕХНІЧНІ ХАРАКТЕРИСТИКИ

2.1 Постановка задачі на розробку програми

Програма повинна мати такі функції:

– створення нестандартного вінка;

– створення пунктів меню;

– інформація про автора;

– виконання необхідних розрахунків.

2.2 Розробка головної програми отримання матриці станів

Головна програма PolPx4mod3 призначена для отримання поліномів ступені P(x) = 4 та їх матриць станів у кінцевому полі GF(3).

Ці обчислення виконуються завдяки повному перебору всіх коефіцієнтів поліномів.

Начальним станом обчислення поліномів є те, що нульовий коефіцієнт та коефіцієнт при старшому степені завжди повинні бути нерівними нулю. Це необхідно через те, що в протилежному випадку такі поліноми вже не відповідатимуть початковим умовам.

Насамперед, поліноми повинні бути характеристичними, а це означає, що повинен бути вільний член полінома.

Блоки 2–4 (додаток А) відповідають за підготування даних для програми.

У блоках 7–10 відбувається отримання дескриптора програми, ресурсів програми у вигляді курсора, іконки програми.

Завантаження ресурсів програми виконується в блоках 8 та 9.

За реєстрацію класу відповідає блок 11. За другий етап створення вікна відповідає функція (процедура) RegisterClassEx, яка створений клас вікна робить доступним програмі, щоб та змогла за його прикладом створювати справжні вікна.

Отримання ширини та висоти вікна відбувається в блоках 13 та 14.

За створення вікна (блок 15) відповідають дві функції: CreateWindowEx – для створення вікна та ShowWindow – для його виведення на екран (блок 16).

Обробка повідомлень виконується в блоках 17–21.

Повідомлення – це структура даних, яка містить елементи, такі, як дескриптор вікна, якому адресоване повідомлення; код (тип) повідомлення та додаткову інформацію.

Функція GetMessage приймає повідомлення, призначене для даної програми, а DispatchMessage відправляє його функції WndProc, яка обслуговує конкретне вікно. У простому випадку така функція реагує тільки на повідомлення WM_DESTROY, яке свідчить про те, що вікно, яке його відправило, в даний момент знищується (наприклад, через натиснення “хрестика” на вікні).

У блоці 22 виконується завершення програми за допомогою функції ExitProcess, яка повертає керування ОС Windows та визволяє ресурси програми.

2.3 Розробка процедури WndProc

Починається процедура з блоків 1 та 2. У блоці 2 відбувається резервування стека під структури: хендл вікна; перерисовування; параметра вікна.

Процедура WndProc необхідна для обслуговування вікна. У цій процедурі оброблюються всі повідомлення програми, як системні повідомлення, так й повідомлення від кнопок меню користувача програми.

Функція WndProc визначає існування вікна. Якщо воно існує, то повідомлення WM_DESTROY не виникає, і всякі інші повідомлення відправляються стандартній функції DefWindowProc, в якій вони і обробляються. Якщо ж вікно знищується, функція PostQuitMessage генерує повідомлення WM_QUIT, яке стає в загальну чергу, а потім передає керування функції GetMessage.

Функція GetMessage відповідає на повідомлення WM_QUIT тим, що повертає нуль в регістр eax. Тому цикл .WHILE припиняється і програма завершує роботу, переходячи до мітки QUIT.

Функція DefWindowProc() передбачає обробку за умовчанням повідомлень, не оброблених у застосуванні.

Для того щоб вивести щось на екран, необхідно перерисувати вікно. Для перерисовування вікна служить повідомлення WM_PAINT. Це повідомлення посилає функція UpdateWindow.

Повідомлення спрямовується безпосередньо віконній процедурі, минаючи чергу повідомлень. Якщо область оновлення порожня, повідомлення не відправляється.

Для того щоб програма могла відобразити на екрані повідомлення, необхідно додати гілку .ELSEIF.

Процедура WndProc викликає процедуру користувача.

У блоці 3 відбувається цикл .IF - .ELSEIF - .ELSE – обробка повідомлень uMsg.

2.4 Розробка процедури користувача PolPx4mod3

Процедура PolPx4mod3 відповідає за формування вигляду поліномів, за розрахунок та виведення матриць станів цих поліномів у файл на жорсткому диску.

У блоці 2 відбувається створення файлу. Запам’ятовування дескриптора fHandle пристрою відбувається в блоці 3.

Мітка m1 є зовнішнім циклом для отримання вигляду полінома, сформовування рядка виведення, обчислення за цим виглядом самої матриці станів та виведення всієї матриці для одного полінома.

У блоці 4 виконується збереження початкового значення в робочих комірках (макрос m2m).

Далі у блоці 5 виконується перетворення формату вигляду полінома за рядком ADDR szSt1,a5,a4,a3,a2,a1.

У блоках 6–9 виконуються функції перетворення змісту першого початкового рядка матриці станів, яке для поліномів за модулем два завжди є 1000.

За міткою mBegin виконується внутрішній цикл, в якому обчислюється вся матриця відповідного полінома. Коли вся матриця порахується та виведеться, тоді відбувається перехід на мітку m1.

У блоках 10–13 відбувається перемноження на коефіцієнти полінома, щоб у блоці 14 отримати значення першої комірки наступного стану СА.

У блоках 15–18 відбувається косий зсув значень комірок.

У блоках 20, 23, 26, 29 виконується порівняння потокового значення з відповідними значеннями стовпця матриці станів.

У блоках 21, 24, 27, 30 виконується збільшення лічильника співпадань. Якщо значення лічильника (блок 31) дорівнює кількості співпадань , то це означає, що виконана генерація повного циклу та ще й одне зайве значення. У такому випадку в блоках 39–41 виконується виведення всіх значень матриці.

Якщо в блоці 31 mm1 ≠ 4, то в блоках 34–37 виконується перетворення отриманих значень.

Очищення буферів від виведених раніше даних використовується в блоці 43.

Для перебору інших (старших) коефіцієнтів полінома використовуються блоки 44–51.

Закриття дескриптора файлу виконується в блоці 52.

Відправлення повідомлення про закінчення обчислень виконується в блоці 53.

2.5 Вхідні дані, робота програми і результати її виконання

Як вхідні дані для програми використовуються:

– завдання степеня полінома задається в пунктах меню;

– максимальна степінь полінома – 4 в полі GF(3);

– збереження результатів роботи програми в файл;

– використати особисту іконку основної програми;

– використання особистої іконки для виведення повідомлень у спрощене віконце;

– використання додаткової кнопки в полі вікна.

Вхідними даними для програми є степінь утворюючого полінома, який задано на етапі проектування та повідомлення, який необхідно обробити й вивести.

У програмі використалися системні повідомлення:

– WM_CLOSE ; якщо знищення вікна та вибір за правою клавішею;
– WM_DESTROY ; обробка повідомлення при знищенні;
– WM_PAINT ; якщо рисування;
– WM_CREATE ; якщо створення;
– WM_COMMAND ; якщо є повідомлення від меню;
– WM_LBUTTONDOWN ; якщо натиснута права кнопка миші.

Повідомлення користувача, які необхідно обробити, викликались у вигляді діалогового вікна (кнопок):

– Button1ID – кнопка "Delete";
– Button2ID – кнопка "Расчеты".

Для роботи програми запустити проект PolPx4mod3.exe.

Після запуску програми виводиться вікно, вигляд якого наведено на рисунку 2.1.

Рисунок 2.1 – Нестандартне головне вікно програми

Для генерації поліномів, їх матриць станів та періоду генерації необхідно натиснути на кнопки “Расчеты”. Після її натискання спочатку буде сформовано файл з даними, потім – з’явиться спливаюче вікно (рисунок 2.2) з назвою “Внимание!!!”.

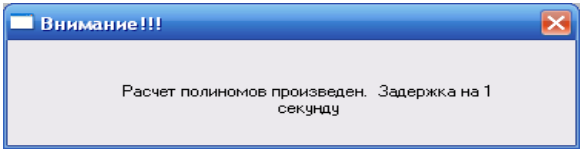


Рисунок 2.2 – Вигляд спливаючого вікна

А потім виведеться повідомлення “Файл с данными и именем PolPx4mod3 находится в каталоге с программой” (рисунок 2.3).

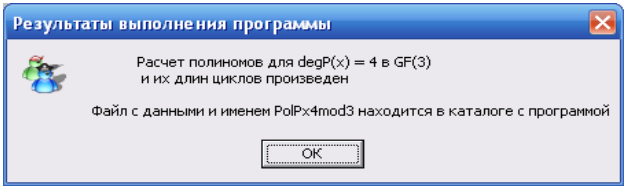


Рисунок 2.3 – Вигляд вікна з повідомленням про виконання розрахунків

Рисунок 2.4 – Додаткове підменю при виборі кнопки “Автор”

При натисканні кнопки “Автор” з’явиться додаткове підменю (рисунок 2.4).

У результаті чого з’явиться повідомлення (рисунок 2.5).

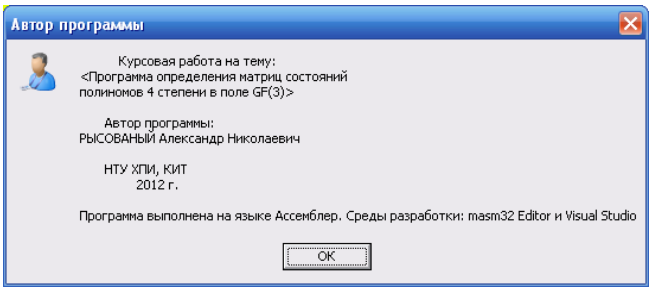


Рисунок 2.5 – Повідомлення при виборі кнопки “Автор”

При виборі кнопки “Справка” з’являється два додаткових підменю (рисунок 2.6).

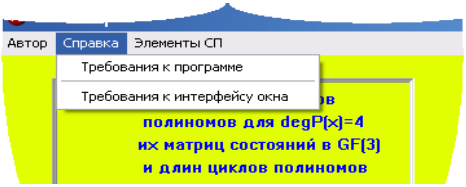


Рисунок 2.6 – Додаткові спливаючі підменю кнопки “Справка”

При виборі кнопки “Требования к программе” з’явиться спрощене вікно з повідомленням (рисунок 2.7).

При виборі кнопки “Требования к интерфейсу окна” з’явиться спрощене вікно з повідомленням (рисунок 2.8).

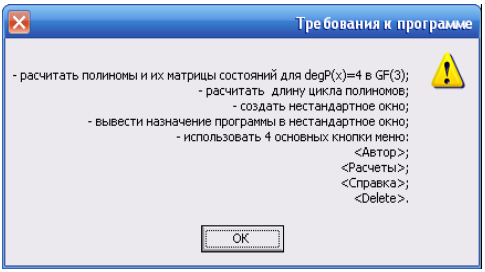


Рисунок 2.7– Повідомлення при виборі кнопки “ Требования к программе”

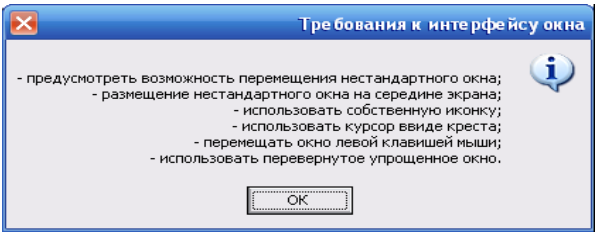


Рисунок 2.8 – Вигляд тексту при виборі кнопки “Требования к интерфейсу окна”

При виборі кнопки “Элементы СП” з’явиться спрощене вікно з повідомленням (рисунок 2.9).

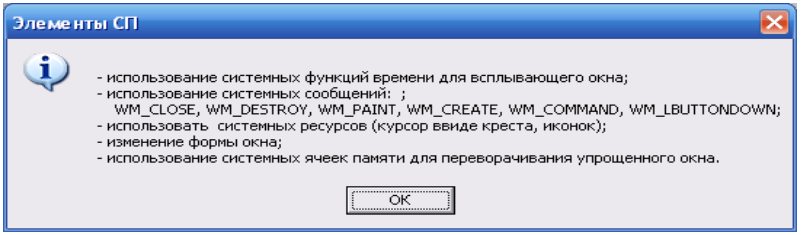


Рисунок 2.9 – Вигляд тексту при виборі кнопки “Элементы СП”

Основне вікно програми знищується при натисканні на кнопку “Delete”, або якщо підвести курсор до полоси вікна з синім кольором та натиснути на праву кнопку миші. У цьому випадку з’явиться вікно (рисунок 2.10).

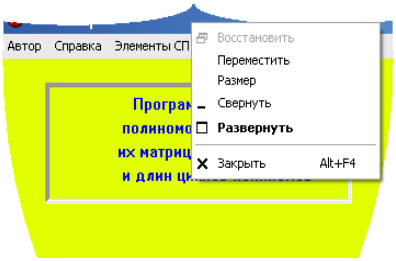


Рисунок 2.10 – Вигляд вікна при натисканні правої кнопки миші

При спробі закрити вікно кнопкою “Закрыть” з’явиться додаткове спрощене вікно (рисунок 2.11).

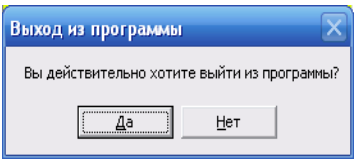


Рисунок 2.11 – Вигляд вікна підтвердження закриття основного вікна програми

Таким чином, програма, яка розглядається, виконує обчислення поліномів, їх матриць станів сигнатурних аналізаторів ступені 4 та кількості циклів генерації.

2.6 Вибір складу технічних і програмних засобів

Функціонування програмного продукту забезпечується стандартною конфігурацією IBM-сумісної ПЕОМ з такими характеристиками:

– CPU-Intel 80286/287 і вище;
– накопичувач на жорсткому магнітному диску;
– відеоадаптер EGA/VGA;
– накопичувач на гнучких магнітних дисках;
– принтер.

Програмне забезпечення, що розробляється, орієнтовано на функціонування під управлінням операційної системи Windows XP.

Текст програми написано мовою низького рівня Асемблер з використанням інтегрованого середовища masm32 v 11.

ВИСНОВКИ

Розроблене в ході курсового проектування ПЗ працює в середовищі Windows XP із застосуванням API-функцій.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Рисований О. М. Системне програмування [Текст]: підручник для студентів напрямку “Компютерна інженерія” вищих навчальних закладів / О.М. Рисований. – Х. : НТУ “ХПІ”, 2010. – 912 с.

2. Кравець В. О. Системне програмування. Програмування на мові асемблера: навч. посіб. / В. О. Кравець, О. М. Рисований. – Х. : НТУ “ХПІ”, 2007. – 448 с.

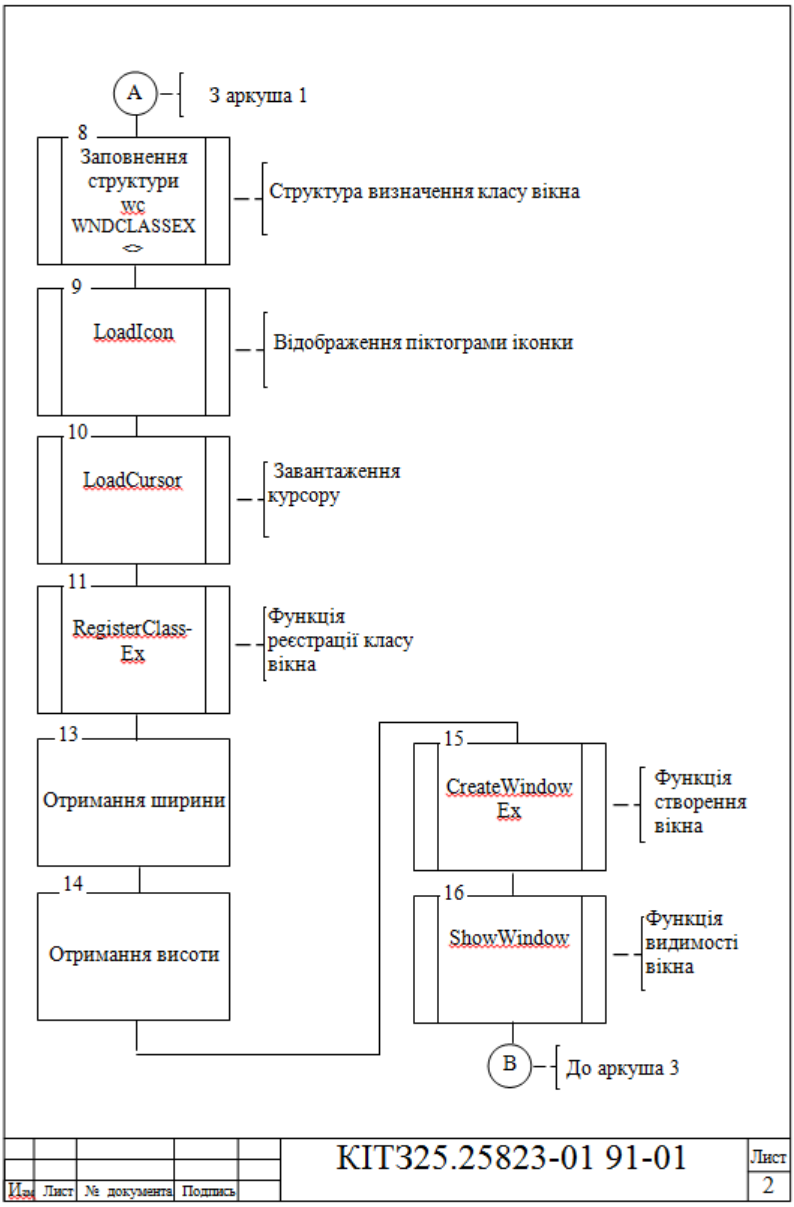
3. Рисований О. М. Цифрові пристрої і мікропроцесори. Архітектура і програмне забезпечення: навч. посіб. / О. М. Рисований, М. В. Грушенко. – Х. : ХУПС, 2005. – 384 с.

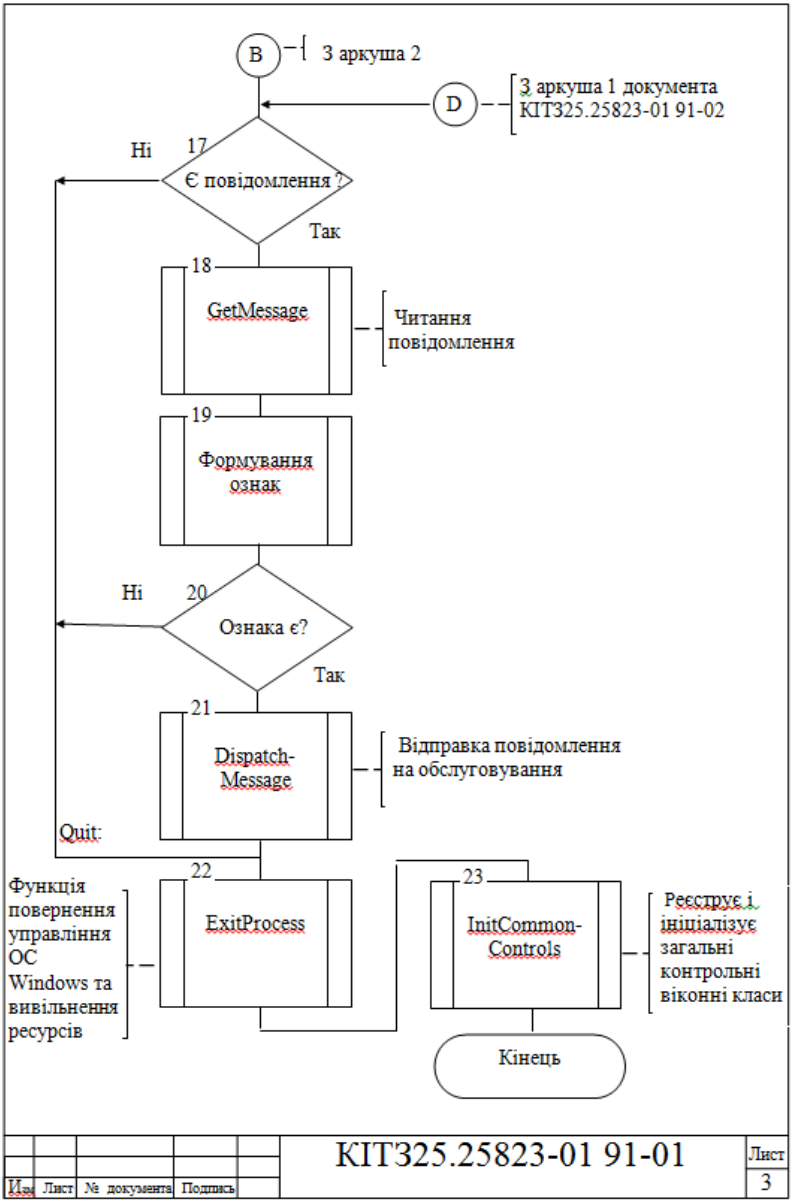
4. Щупак Ю. А. Win32 API. Эффективная разработка приложений / Ю. А. Щупак. – СПб. : Питер, 2007. – 572 с.

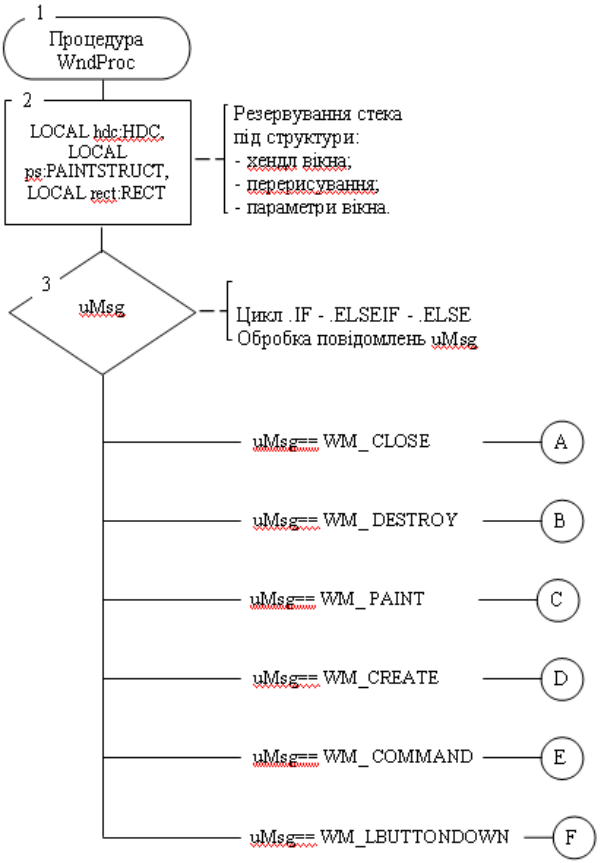
5. Несвижский В. Программирование аппаратных средств в Windows / В. Несвижский. – СПб. : БХВ-Петербург, 2004. – 880 с.

Додаток

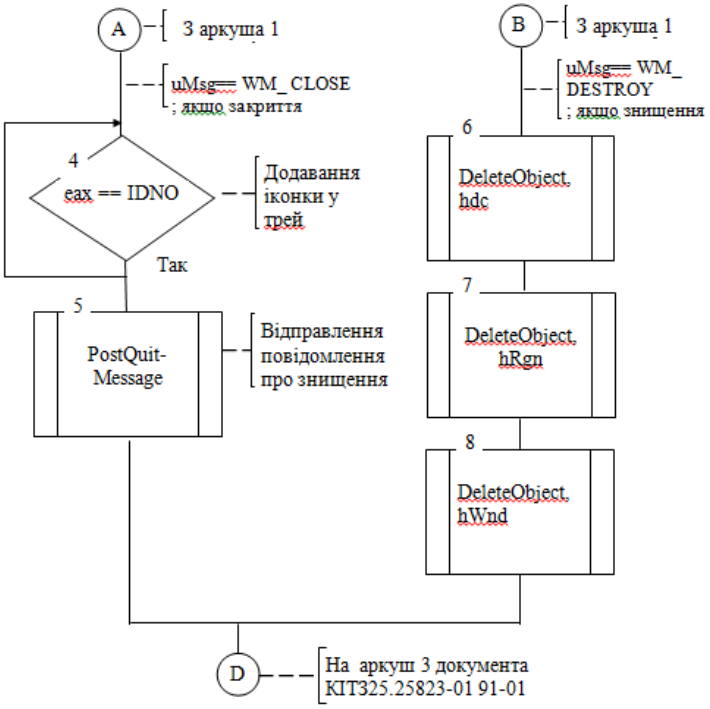
					КІТ325.25823-01 91-01	
Ізм	Лист	№ документу	Підпис	Дата		
Розроб.		Матлашов			Головна програма PolPx4mod3 отримання матриці станів Схема алгоритму	Літ. / Лист 1 / Листів 3
Перев.		Рисований				
Н.						НТУ «ХПІ» Кафедра ОТП
Н. контр.		Рисований				
Затверди		Домнін				


A
З аркуша 1
8
Заповнення структури wc WNDCLASSEX
Структура визначення класу вікна
9
LoadIcon
Відображення піктограми іконки
10
LoadCursor
Завантаження курсору
11
RegisterClass-Ex
Функція реєстрації класу вікна
13
Отримання ширини
14
Отримання висоти
15
CreateWindow Ex
Функція створення вікна
16
ShowWindow
Функція видимості вікна
B
До аркуша 3
КІТ325.25823-01 91-01
Лист
2
Изм
Лист
№ документа
Подпись

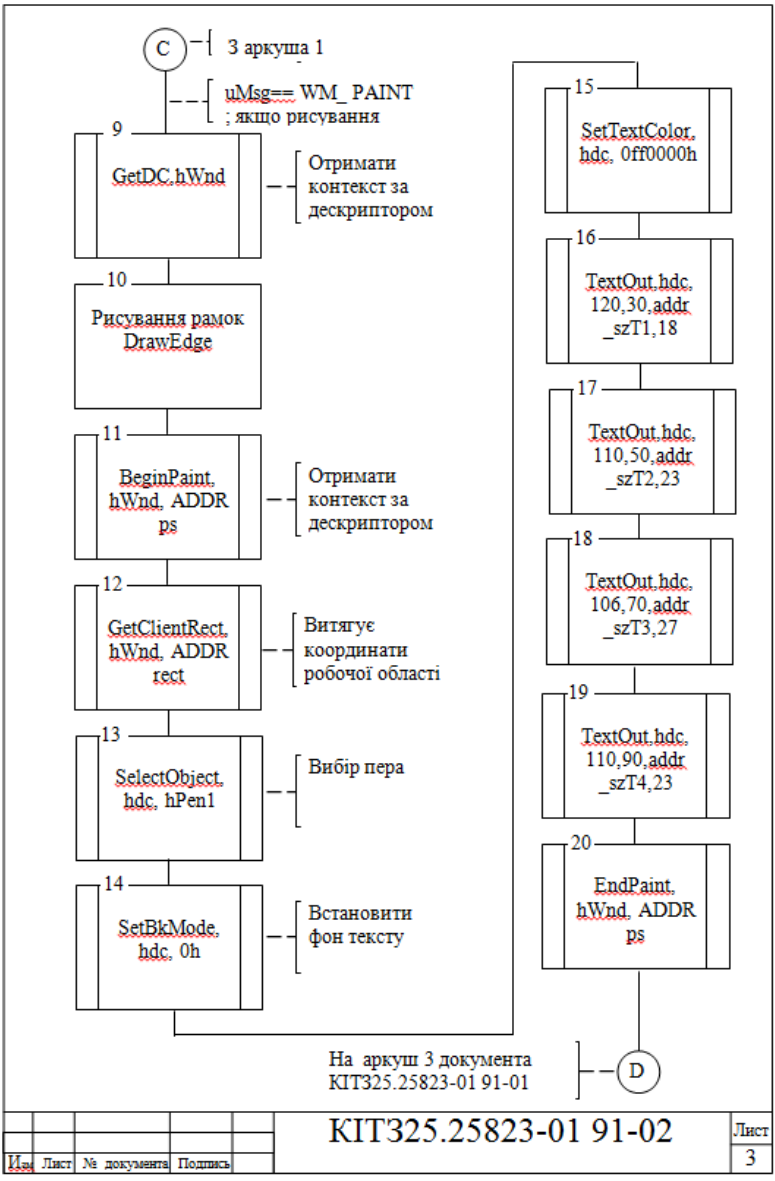

B
З аркуша 2
D
З аркуша 1 документа КІТ325.25823-01 91-02
17
Ні
Є повідомлення?
Так
18
GetMessage
Читання повідомлення
19
Формування ознак
20
Ні
Ознака є?
Так
21
Dispatch-Message
Відправка повідомлення на обслуговування
Quit:
22
ExitProcess
Функція повернення управління ОС Windows та вивільнення ресурсів
23
InitCommon-Controls
Реєструє і ініціалізує загальні контрольні віконні класи
Кінець
КІТ325.25823-01 91-01
Лист
3
Изм
Лист
№ документа
Подпись

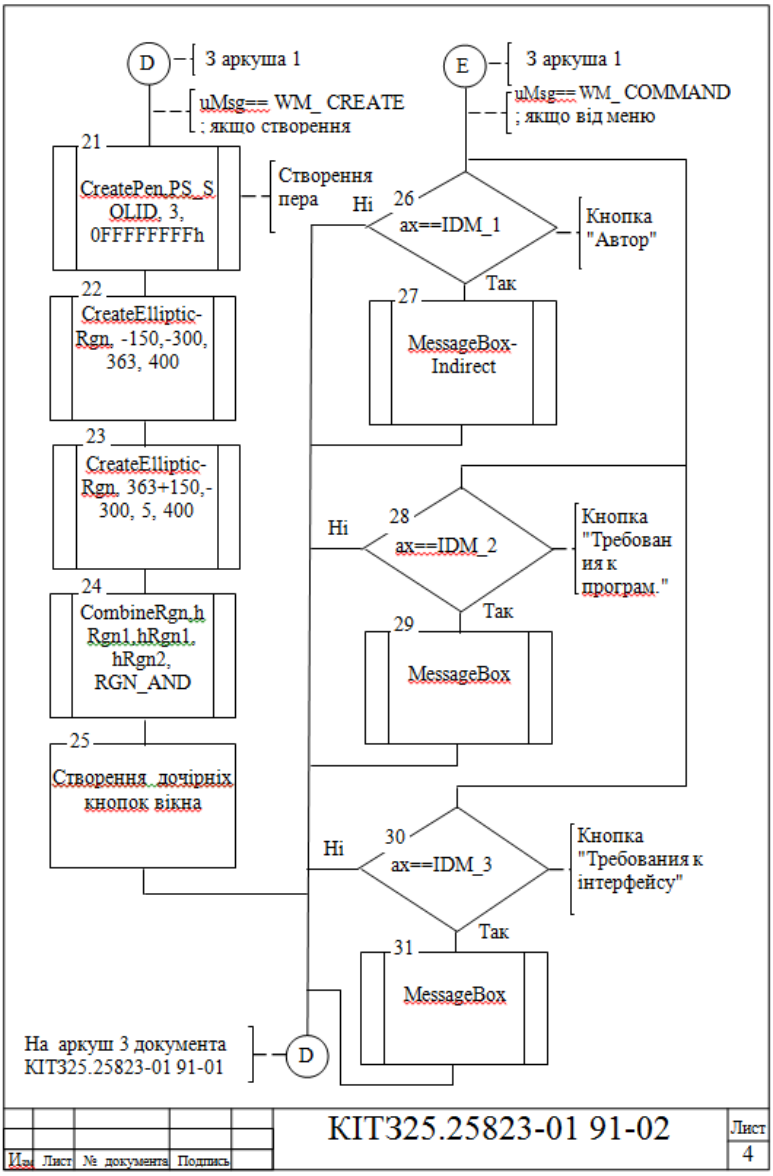


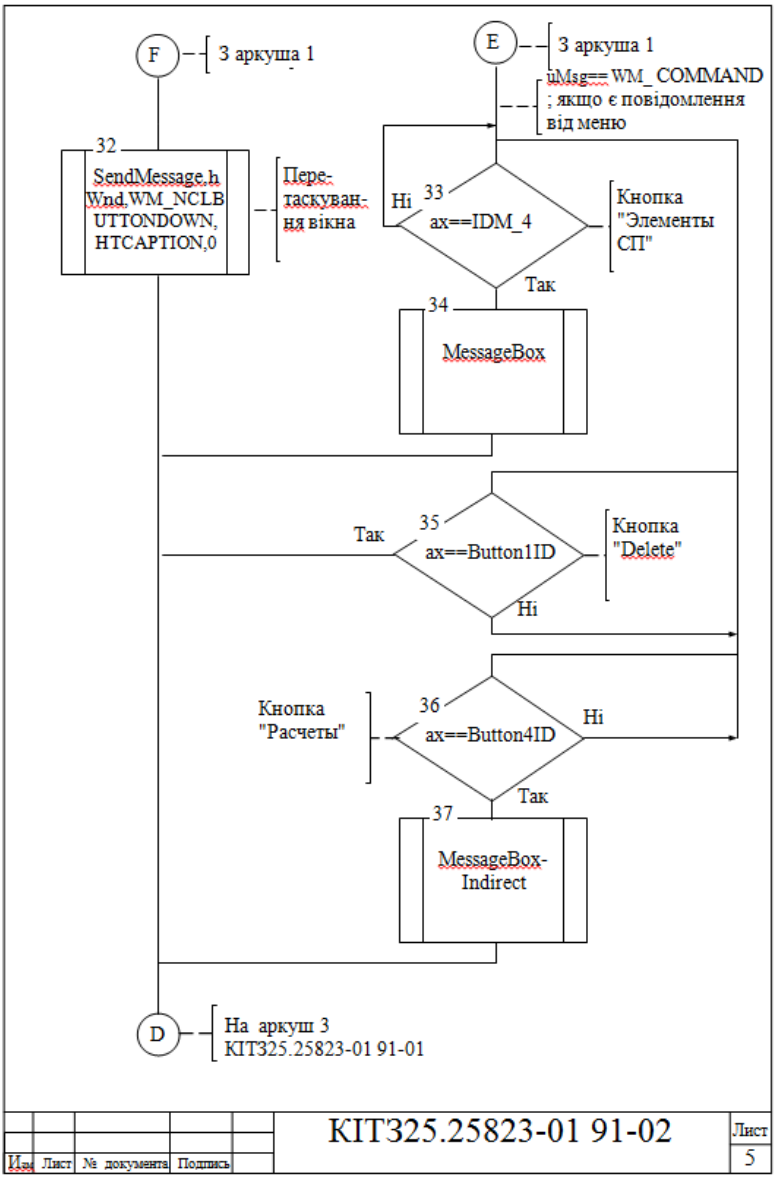
Ізм.	Лист	№ документу	Підпис	Дата	КІТ325.25823-01 91-02		
Розроб.		Матлашов			Процедура WndProc створення вікна та повідомлень користувача Схема алгоритму	Літ.	Лист 1 / Листів 5
Перев.		Рисований					
Н. контр.		Рисований				НТУ «ХПІ» Кафедра ОТП	
Затвердив		Домнін					

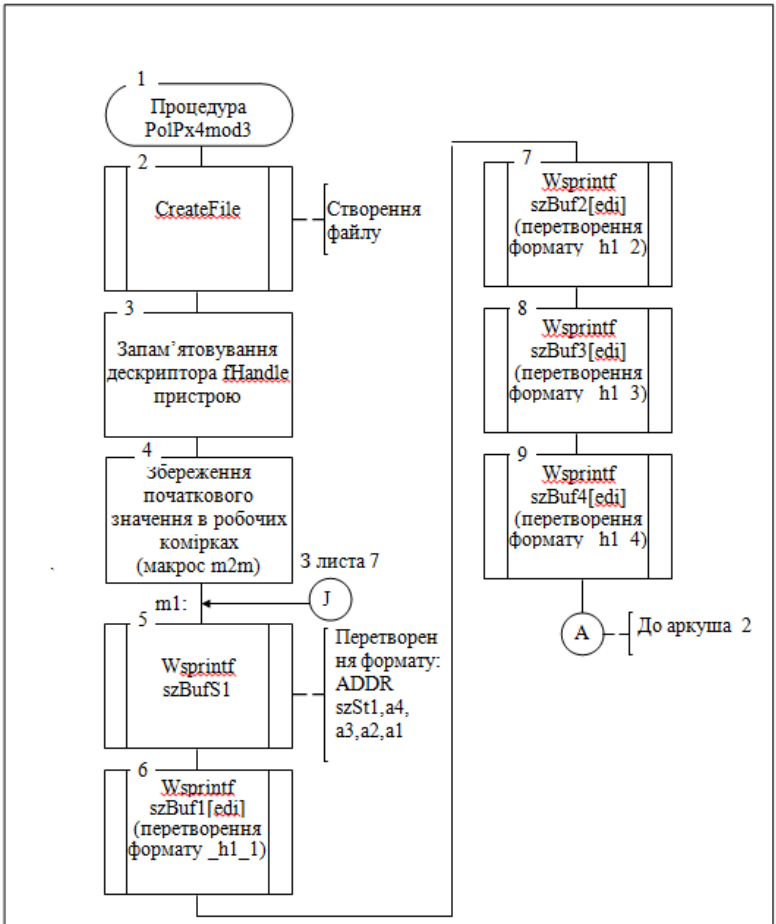


					КІТ325.25823-01 91-02	Лист
						2
Изм	Лист	№ документа	Подпись			


C
З аркуша 1
uMsg== WM_ PAINT
; якщо рисування
9
GetDC,hWnd
Отримати контекст за дескриптором
10
Рисування рамок DrawEdge
11
BeginPaint, hWnd, ADDR ps
Отримати контекст за дескриптором
12
GetClientRect, hWnd, ADDR rect
Витягує координати робочої області
13
SelectObject, hdc, hPen1
Вибір пера
14
SetBkMode, hdc, 0h
Встановити фон тексту
15
SetTextColor, hdc, 0ff0000h
16
TextOut,hdc, 120,30,addr _szT1,18
17
TextOut,hdc, 110,50,addr _szT2,23
18
TextOut,hdc, 106,70,addr _szT3,27
19
TextOut,hdc, 110,90,addr _szT4,23
20
EndPaint, hWnd, ADDR ps
На аркуш 3 документа КІТ325.25823-01 91-01
D
Изм Лист № документа Подпись
КІТ325.25823-01 91-02
Лист 3

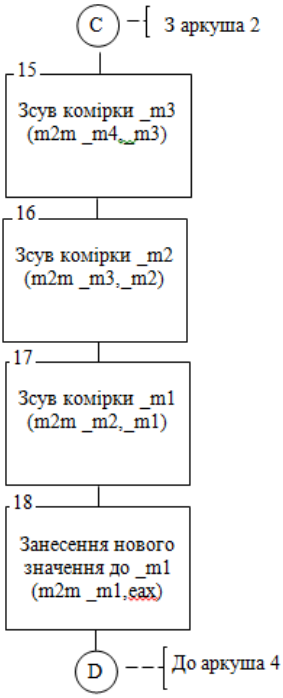

D
З аркуша 1
uMsg== WM_ CREATE
; якщо створення
E
З аркуша 1
uMsg== WM_ COMMAND
; якщо від меню
21
CreatePen,PS_ SOLID, 3, 0FFFFFFFFh
Створення пера
Ні
26
ax==IDM_1
Кнопка "Автор"
Так
27
MessageBox-Indirect
22
CreateElliptic-Rgn, -150,-300, 363, 400
23
CreateElliptic-Rgn, 363+150,-300, 5, 400
Ні
28
ax==IDM_2
Кнопка "Требован ия к програм."
Так
29
MessageBox
24
CombineRgn,hRgn1,hRgn1, hRgn2, RGN_AND
25
Створення дочірніх кнопок вікна
Ні
30
ax==IDM_3
Кнопка "Требования к інтерфейсу"
Так
31
MessageBox
На аркуш 3 документа КІТ325.25823-01 91-01
D
КІТ325.25823-01 91-02
Лист
4
Изм
Лист
№ документа
Подпись


F
З аркуша 1
E
З аркуша 1
uMsg== WM_ COMMAND ; якщо є повідомлення від меню
32
SendMessage,hWnd,WM_NCLBUTTONDOWN, HTCAPTION,0
Пере-таскування вікна
Ні
33
ax==IDM_4
Кнопка "Элементы СП"
Так
34
MessageBox
Так
35
ax==Button1ID
Кнопка "Delete"
Ні
Кнопка "Расчеты"
36
ax==Button4ID
Ні
Так
37
MessageBox-Indirect
D
На аркуш 3
КІТ325.25823-01 91-01
Изм. Лист № документа Подпись
КІТ325.25823-01 91-02
Лист
5

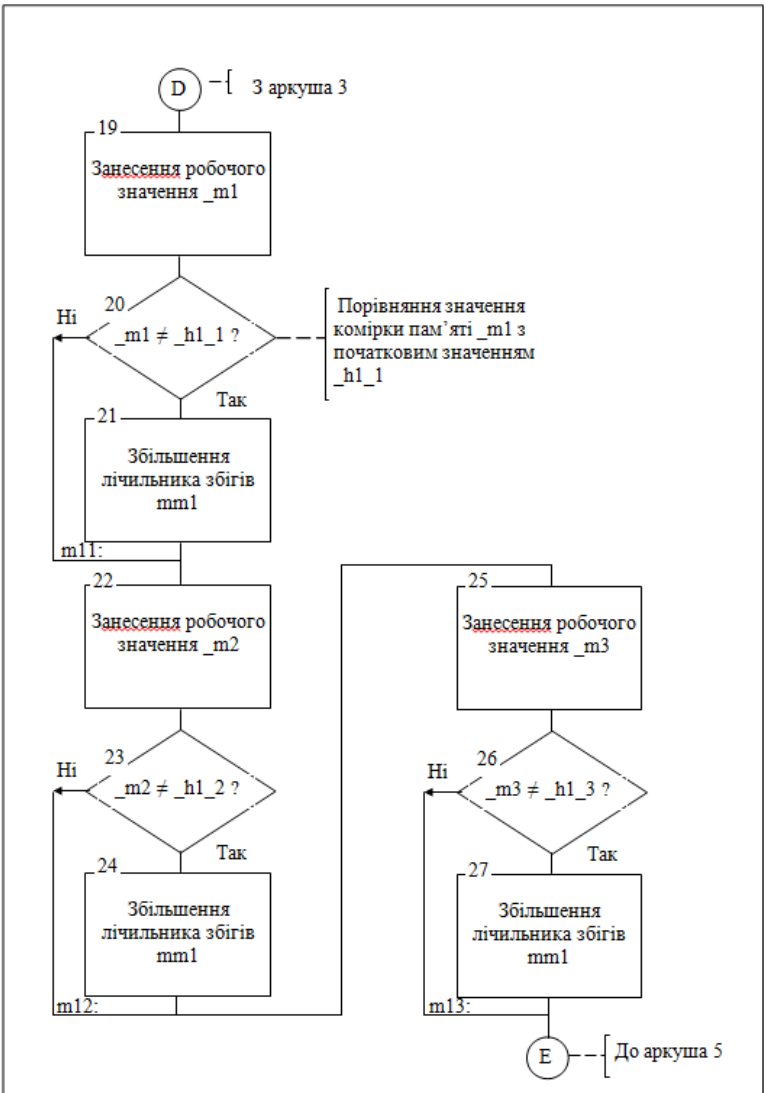


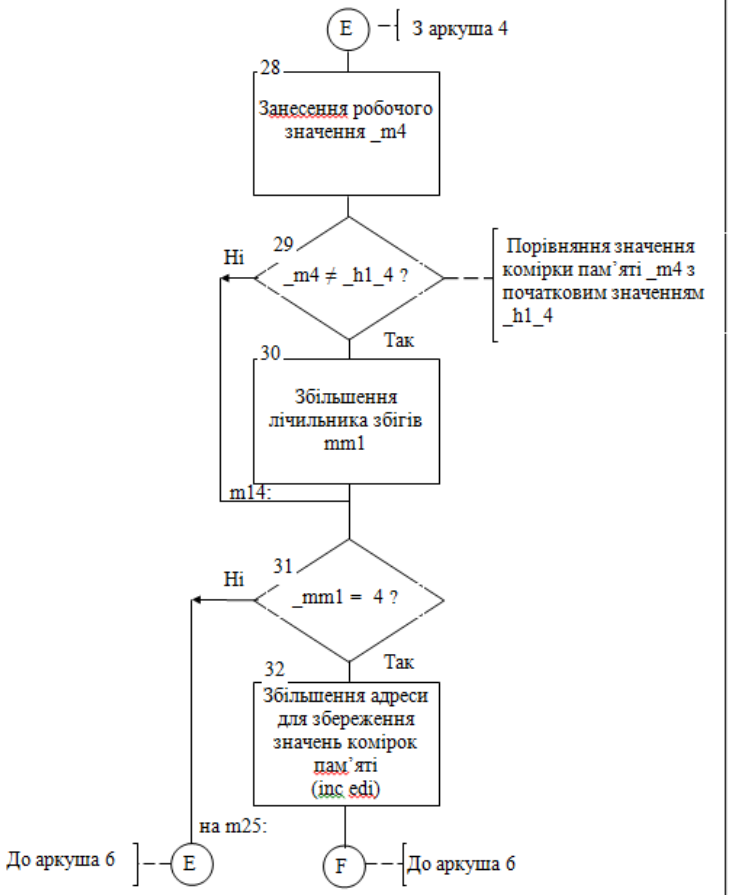
					КІТ325.25823-01 91-03	
Ізм	Лист	№ документу	Підпис	Дата		
Розроб.		Матлащов			Процедура PolPx4mod3 Proc отримання матриці станів Схема алгоритму	Літ. / Лист 1 / Листів 8
Перев.		Рисований				
Н. п/отд.						НТУ «ХПІ» Кафедра ОТП
Н. контр.		Рисований				
Затвердив		Домнін				

					КІТ325.25823-01 91-03	Лист
Изм	Лист	№ документа	Подпись			2

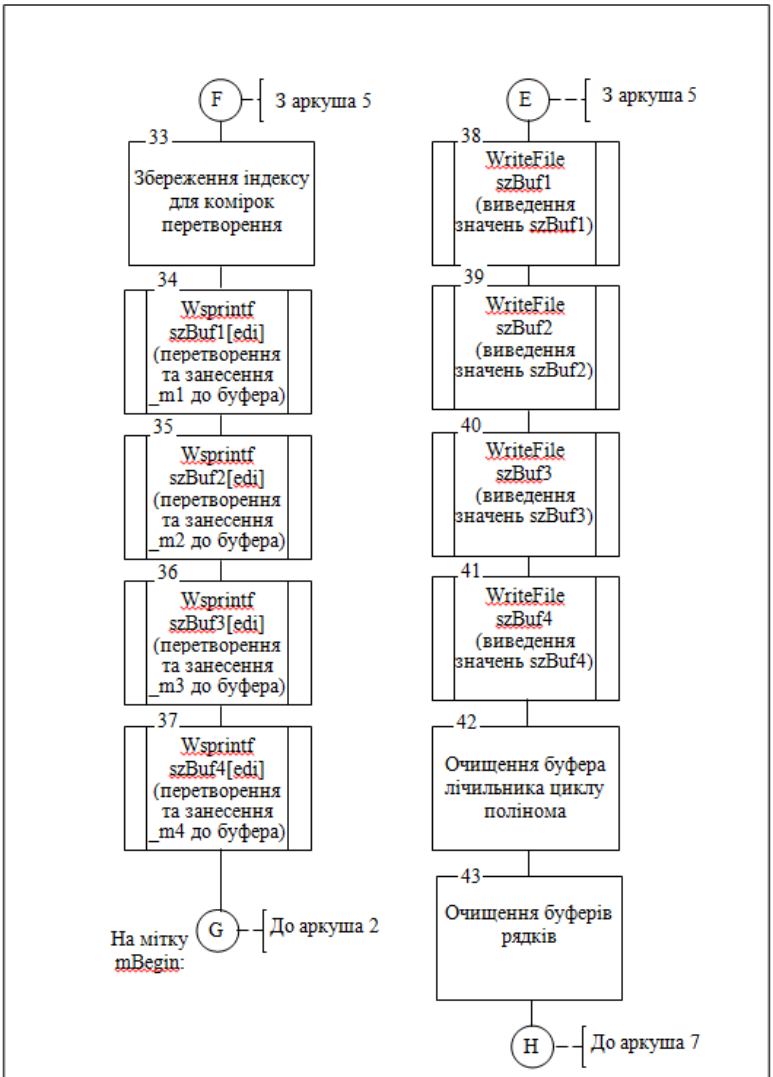


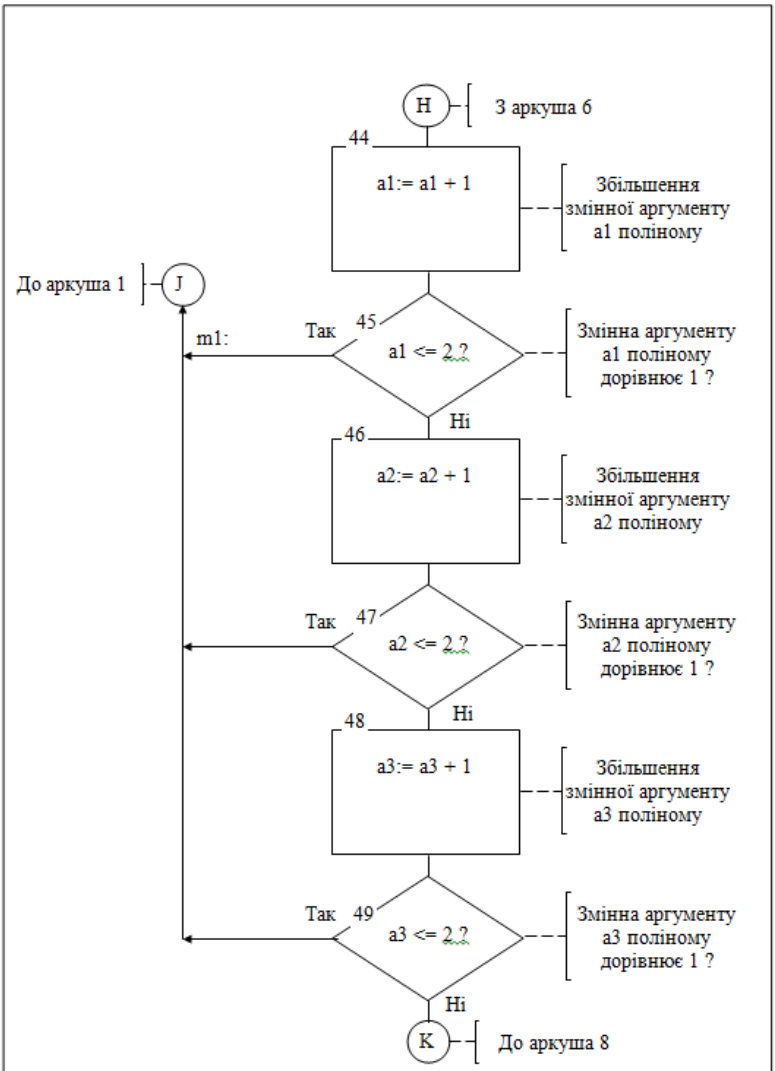
					КІТ325.25823-01 91-03	Лист
Изм.	Лист	№ документа	Подпись			3

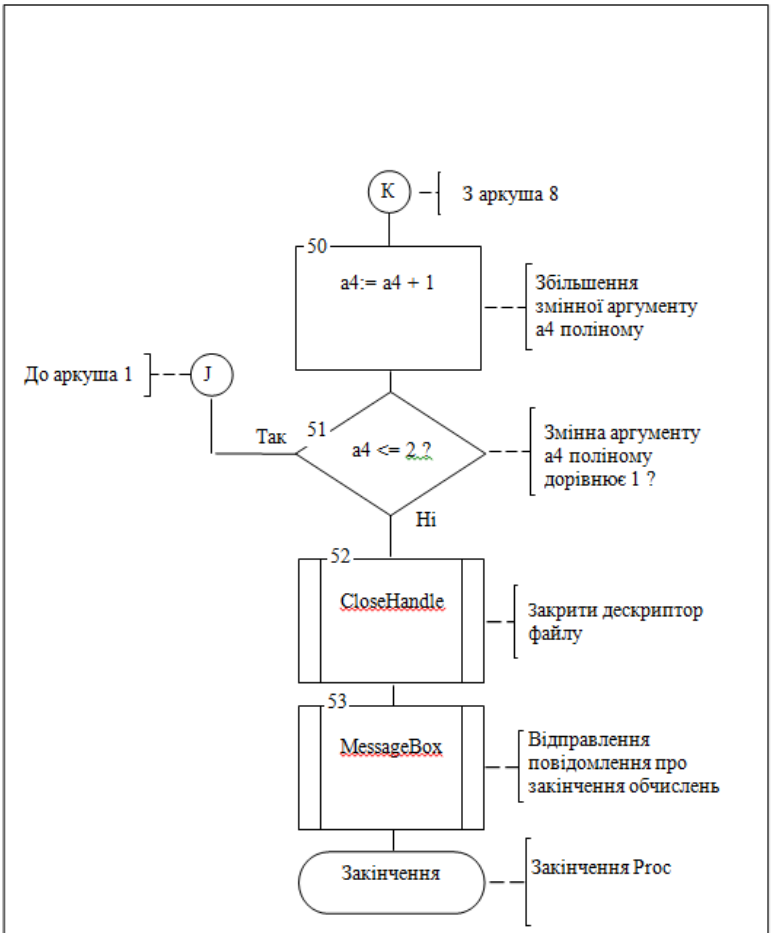

D
З аркуша 3
19
Занесення робочого значення _m1
20
_m1 ≠ _h1_1 ?
Ні
Так
Порівняння значення комірки пам'яті _m1 з початковим значенням _h1_1
21
Збільшення лічильника збігів mm1
m11:
22
Занесення робочого значення _m2
23
_m2 ≠ _h1_2 ?
Ні
Так
24
Збільшення лічильника збігів mm1
m12:
25
Занесення робочого значення _m3
26
_m3 ≠ _h1_3 ?
Ні
Так
27
Збільшення лічильника збігів mm1
m13:
E
До аркуша 5
КІТ325.25823-01 91-03
Лист
4
Изм
Лист
№ документа
Подпись



					КІТ325.25823-01 91-03	Лист
Изм.	Лист	№ документа	Подпись			5


F
З аркуша 5
33
Збереження індексу для комірок перетворення
34
Wsprintf szBuf1[edi] (перетворення та занесення _m1 до буфера)
35
Wsprintf szBuf2[edi] (перетворення та занесення _m2 до буфера)
36
Wsprintf szBuf3[edi] (перетворення та занесення _m3 до буфера)
37
Wsprintf szBuf4[edi] (перетворення та занесення _m4 до буфера)
На мітку mBegin:
G
До аркуша 2
E
З аркуша 5
38
WriteFile szBuf1 (виведення значень szBuf1)
39
WriteFile szBuf2 (виведення значень szBuf2)
40
WriteFile szBuf3 (виведення значень szBuf3)
41
WriteFile szBuf4 (виведення значень szBuf4)
42
Очищення буфера лічильника циклу полінома
43
Очищення буферів рядків
H
До аркуша 7
Изм
Лист
№ документа
Подпись
КІТ325.25823-01 91-03
Лист
6


H
З аркуша 6
44
a1:= a1 + 1
Збільшення змінної аргументу a1 поліному
До аркуша 1
J
m1:
Так
45
a1 <= 2 ?
Змінна аргументу a1 поліному дорівнює 1 ?
Ні
46
a2:= a2 + 1
Збільшення змінної аргументу a2 поліному
Так
47
a2 <= 2 ?
Змінна аргументу a2 поліному дорівнює 1 ?
Ні
48
a3:= a3 + 1
Збільшення змінної аргументу a3 поліному
Так
49
a3 <= 2 ?
Змінна аргументу a3 поліному дорівнює 1 ?
Ні
K
До аркуша 8
Изм
Лист
№ документа
Подпись
КІТ325.25823-01 91-03
Лист
7



Изм.	Лист	№ документа	Подпись		КІТ325.25823-01 91-03	Лист 8

Додаток Д

Зразок оформлення тексту прорами

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРЇНИ
Національний технічний університет
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра обчислювальної техніки та програмування

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТП
_____ / Домнін Ф.А. /
“____” _____ 2013 р.

ПРОГРАМА ВИЗНАЧЕННЯ МАТРИЦЬ СТАНІВ ПОЛІНОМІВ
4-ГО СТЕПЕНЯ В ПОЛІ GF(3)

Текст програми

АРКУШ ЗАТВЕРДЖЕННЯ

КІТ00А.077777-01 12 01

Розробники

Керівник проекту
доц. каф. ОТП
_____ к.т.н., доц. Рисований О.М.

Виконавець
студент групи КІТ00А
_____ Іванов І. І.
“____” _____ 2013 р.

Харків 2013

ЗАТВЕРДЖЕНО
КІТ00А.077777-01 12 01

ПРОГРАМА ВИЗНАЧЕННЯ МАТРИЦЬ СТАНІВ ПОЛІНОМІВ 4-ГО СТЕПЕНЯ В ПОЛІ GF(3)

Текст програми

КІТ00А.077777-01 12 01
Аркушів 16

Харків 2013

АНОТАЦІЯ

Поданий вміст файлів з програмними кодами програми розрахунку станів поліномів 4-го степеня для подальшого використання при побудові паралельного сигнатурного аналізатора. Розробка виконана на мовах програмування асемблер masm32 і C++ в середовищах masm32 Editor і Visual Studio 2010.

АННОТАЦИЯ

Представлено содержимое файлов с программными кодами программы расчета состояний полиномов 4-й степени для дальнейшего использования при построении параллельного сигнатурного анализатора. Разработка выполнена на языках программирования ассемблер masm32 и C++ в средах masm32 Editor и Visual Studio 2010.

THE ANNOTATION

Content of files is presented with the programmatic codes of the program of calculation of the states of polynomials of 4th degree for the further using for the construction of parallel signaturnogo analyzer. Development is executed on programming languages assembler of masm32 and C++ in the environments of masm32 Editor and Visual Studio 2010.

ЗМІСТ

1 Головна програма KURS

```
; Найменування програми – “Отримання матриці станів”
; Автор – Рисований Олександр Миколайович
; Дата створення програми – 15.05.2012 року
; Номер версії – 4.55
.686 ; директива визначення типу мікропроцесора
.model flat,stdcall ; завдання лінійної моделі пам’яті та угоди ОС Windows
option casemap:none ; відмінність малих та великих літер
include \masm32\include\windows.inc ; файли структур, констант …
include \masm32\macros\macros.asm
uselib masm32, user32, kernel32,gdi32, shell32,comctl32
IDI_ICON equ 22
IDI_ICON2 equ 33
IDI_ICON3 equ 44
IDM_1 equ 1 ; ідентифікатор кнопки меню "Автор программы"
IDM_2 equ 2 ; ідентифікатор кнопки меню "Требования к программе"
IDM_3 equ 3 ; ідентифікатор кнопки меню "Требования к интерфейсу окна"
IDM_4 equ 4 ; ідентифікатор кнопки меню "Элементы СП"
Button1ID equ 5 ; ідентифікатор кнопки "Delete"
Button4ID equ 6 ; ідентифікатор кнопки "Расчеты"
DialogProc proto :HWND, :UINT, :WPARAM, :LPARAM
IDM_POPUP equ 37 ; ідентифікатор повідомлення від діалогового вікна
BSIZE equ 160 ; !!!кількість байтів, які записуються у файл
.data
_szT1 db "Программа расчетов",0
_szT2 db "полиномов для degP(x)=4,",0
_szT3 db "их матриц состояний в GF(3)",0
_szT4 db "и длин циклов полиномов",0
hwndButton HWND ?
ButtonClassName db "button",0
ButtonText1 db "Delete",0
ButtonText4 db "Расчеты",0
info db "Вы действительно хотите выйти из программы?",0
_Title db "Выход из программы",0
szTitle1 db "Автор программы",0
Msg1 db " Курсовая работа на тему:",0ah,0dh,\
"<Программа определения матриц состояний",0ah,0dh,\
"полиномов 4 степени в поле GF(3)>",0ah,0dh,0dh,\
" Автор программы:",0ah,0dh,\
"РЫСОВАНЫЙ Александр Николаевич ",0ah,0dh,0dh," НТУ ХПИ, КИТ ",0ah,0dh,\
" 2012 г.",0ah,0dh,0dh,\
"Программа выполнена на языке Ассемблер. Среды разработки: masm32 Editor и
Visual Studio",0
szTitle2 db "Требования к программе",0 ;
szInf2 db 0dh,"- рассчитать полиномы и их матрицы состояний для degP(x)=4 в
GF(3);",0ah,0dh,\ ;
"- рассчитать длину цикла полиномов;",0ah,0dh,\ ;
```

```
"- создать нестандартное окно;",0ah,0dh,\ ;
"- вывести назначение программы в нестандартное окно;",0ah,0dh,\ ;
"- использовать 4 основных кнопки меню:",0ah,0dh,\ ;
"        <Автор>;",0ah,0dh,\
"       <Расчеты>;",0ah,0dh,\
"        <Справка>;",0ah,0dh,\
"        <Delete>.",0

szTitle3 db "Требования к интерфейсу окна",0 ;
szInf3 db 0dh,"- предусмотреть возможность перемещения нестандартного
окна;",0ah,0dh,\
"- размещение нестандартного окна на середине экрана;",0ah,0dh,\
"- использовать собственную иконку;",0ah,0dh,\
"- использовать курсор в виде креста;",0ah,0dh,\
"- перемещать окно левой клавишей мыши;",0ah,0dh,\
"- использовать перевернутое упрощенное окно.",0

szTitle4 db "Элементы СП",0 ;
szInf4 db 0dh,"- использование системных функций времени для всплывающего
окна;",0ah,0dh,\
"- использование системных сообщений:  ;",0ah,0dh,\
"        WM_CLOSE, WM_DESTROY, WM_PAINT, WM_CREATE, WM_COMMAND,
WM_LBUTTONDOWN;",0ah,0dh,\
"- использовать  системных ресурсов (курсор ввиде креста, иконок);",0ah,0dh,\
"- изменение формы окна;",0ah,0dh,\
"- использование системных ячеек памяти для переворачивания упрощенного
окна.",0
ClassName db "SimpleWinClass",0
AppName db "Роза ветров",0
MenuName db "FirstMenu",0
 winWid dd 700
 winHgt dd 700
 left   dd ?
 top    dd ?
   spt1 RECT <40,20,320,122>     ; координати рамки 1
   spt2 RECT <42,22,318,120>     ; координати рамки 2
  wc WNDCLASSEX <>               ;ініціалізація
msg MSG <>
params MSGBOXPARAMS <>
hwnd HWND ?
hRgn dd ?
hInstance HINSTANCE ?
hPen1 dd 0

Titl2 db "Результаты выполнения программы",0
Msg2 db "          Расчет полиномов для degP(x) = 4 в GF(3)",0ah,0dh,\
"          и их длин циклов произведен",0ah,0dh,0dh,\
"Файл с данными и именем PolPx4mod3 находится в каталоге с программой",0
```

```
hRgn1 dd ?
hRgn2 dd ?

.code
start:
invoke GetModuleHandle, NULL   ; отримання дескриптора програми
 mov   hInstance,eax           ; збереження дескриптора програми
 mov   wc.cbSize,SIZEOF WNDCLASSEX      ; кількість байтів структури
 mov   wc.style, CS_HREDRAW or CS_VREDRAW ; стиль та поведінка вікна
 mov   wc.lpfnWndProc, OFFSET WndProc           ; адреса процедури WndProc
 mov   wc.cbClsExtra,NULL     ; кількість байтів для структури
 mov   wc.cbWndExtra,NULL   ; кількість байтів для структури
 m2m wc.hInstance,hInstance ;запис дескриптора програми в поле структури
invoke  GetStockObject,WHITE_BRUSH ; Створення кольору
invoke CreateSolidBrush,0000FFE1h  ;  заливки
 mov wc.hbrBackground,eax         ;  вікна
 mov wc.lpszMenuName,OFFSET MenuName   ; ім'я ресурсу меню
 mov wc.lpszClassName,OFFSET ClassName ; ім'я класу
invoke LoadIcon,NULL,IDI_ERROR      ; іконка
 mov wc.hIcon,eax
 mov wc.hIconSm,eax
invoke LoadCursor,NULL,IDC_CROSS    ; курсор – хрест
 mov wc.hCursor,eax
invoke RegisterClassEx,addr wc
invoke GetSystemMetrics,SM_CXSCREEN ; отримання ширини
 shr eax,1            ; знаходження центра
 mov ebx, winWid
    shr ebx, 1
    sub eax, ebx
    mov left, eax
invoke GetSystemMetrics,SM_CYSCREEN ; отримання висоти
 shr eax,1            ; знаходження центра
 mov ebx, winHgt
    shr ebx, 1
    sub eax, ebx
    mov top, eax
invoke          CreateWindowEx,0,ADDR          ClassName,ADDR          AppName,
WS_OVERLAPPEDWINDOW, left,top,winWid,winHgt,0,0,hInstance,0 ; гориз. та верт.
координати, ширина й висота
  mov hwnd,eax              ;збереження дескриптора
invoke ShowWindow,hwnd,SW_SHOWNORMAL ;показати вікно
invoke UpdateWindow,hwnd           ;відновити вікно

.WHILE TRUE ; обробка повідомлень від окна
  invoke GetMessage,ADDR msg,0,0,0 ; читання повідомлення
    or eax,eax
```

```
    jz Quit
    invoke DispatchMessage,ADDR msg ; відправка на обслуговування
.ENDW
invoke DestroyWindow,hwnd
Quit:
mov eax,msg.wParam       ;очищення ресурсів
invoke ExitProcess,eax    ;
invoke InitCommonControls ; реєструє і ініціалізує загальні контрольні віконні класи

WndProc proc hWnd:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM
LOCAL hdc:HDC ;контекст
LOCAL ps:PAINTSTRUCT ;структура для рисування
LOCAL rect:RECT      ;структура координат
mov eax,wParam       ;завантаження wParam
.IF uMsg == WM_CLOSE    ; якщо знищення вікна та вибів за прав. клав.
     invoke MessageBox,0,offset info,offset _Title,MB_YESNO
     .if eax == IDNO
      jmp @wmpaind
     .endif
    invoke PostQuitMessage, NULL ; відправлення повідомлення про знищення вікна
.ELSEIF uMsg ==WM_DESTROY ; обробка  повідомлення при знищенні
    invoke DeleteObject, hdc ; видаляє з пам'яті і звільняє ресурси
    invoke DeleteObject, hRgn ; видаляє з пам'яті і звільняє ресурси
     invoke DestroyWindow,hWnd  ; видалення вікна
     ret
.ELSEIF uMsg==WM_PAINT   ; якщо рисування
@wmpaind:
  invoke GetDC,hWnd  ; отримати контекст за дескриптором
    mov hdc,eax        ; зберегти
  invoke DrawEdge,hdc,ADDR spt1,EDGE_SUNKEN,BF_RECT; рисування рамки spt1
  invoke DrawEdge,hdc,ADDR spt2,EDGE_SUNKEN,BF_RECT; рисування рамки spt2
  invoke BeginPaint,hWnd, ADDR ps ; початок рисування
   mov   hdc,eax
invoke GetClientRect,hWnd, ADDR rect ; витягує координати робочої області вікна
invoke SelectObject,hdc,hPen1    ;  вибір пера
  invoke SetBkMode,hdc,0h       ; встановити фон тексту
  invoke SetTextColor,hdc,0ff0000h ; встановлення кольору тексту
  invoke TextOut,hdc,120,30,addr _szT1,18 ; виведення тексту
  invoke TextOut,hdc,110,50,addr _szT2,23 ; виведення тексту
  invoke TextOut,hdc,106,70,addr _szT3,27 ; виведення тексту
  invoke TextOut,hdc,110,90,addr _szT4,23 ; виведення тексту
  invoke EndPaint,hWnd, ADDR ps   ;завершення рисування
.ELSEIF uMsg==WM_CREATE      ; якщо створення
  invoke CreatePen,PS_SOLID,3,0FFFFFFFFh ; створення пера
  mov hPen1,eax                      ; збереження пера
; перетинання двох еліпсів
invoke CreateEllipticRgn,-150,-300,363,400 ; видима права велика частина еліпса
  mov hRgn1, eax;
```

```
invoke CreateEllipticRgn,363+150,-300,5,400 ; видима ліва велика частина еліпса
  mov hRgn2,eax;
invoke CombineRgn,hRgn1,hRgn1,hRgn2,RGN_AND ; накладення двох еліпсів
invoke  CreateEllipticRgn,-40,-15,180,23 ; верхня ліва частина еліпса
  mov hRgn2,eax;
invoke  CombineRgn,hRgn1,hRgn1,hRgn2,RGN_DIFF  ;  віднімаємо  верхню  ліву
частину
;; відрізання зверху праворуч
 invoke CreateEllipticRgn,180,-15,363+40,23 ; верхня права частина еліпса
  mov hRgn2,eax ;
 invoke  CombineRgn,hRgn1,hRgn1,hRgn2,RGN_DIFF;
 invoke SetWindowRgn,hWnd,hRgn1,1    ; призначаємо отриманий регіон формі

 ;;Створення  дочірніх кнопок вікна hWnd
invoke CreateWindowEx,0, ADDR ButtonClassName,ADDR ButtonText1,\ ; "Delete"
        WS_CHILD or WS_VISIBLE or BS_DEFPUSHBUTTON,\
            153,280,50,23,hWnd,Button1ID,hInstance,0
invoke CreateWindowEx,0, ADDR ButtonClassName,ADDR ButtonText4,\ ; "Расчеты"
        WS_CHILD or WS_VISIBLE or BS_DEFPUSHBUTTON,\
            135,240,84,23,hWnd,Button4ID,hInstance,0
       mov hwndButton,eax

.ELSEIF uMsg==WM_COMMAND    ; якщо є повідомлення від меню
mov eax,wParam
  .IF ax==IDM_1       ; якщо вибрана кнопка меню "Автор"
      mov params.cbSize,SIZEOF MSGBOXPARAMS  ; розмір структури
            mov params.hwndOwner, 0           ; дескриптор вікна власника
   invoke GetModuleHandle, 0              ; отримання дескриптора програми
            mov params.hInstance, eax          ; збереження дескриптора програми
            mov params.lpszText, offset Msg1   ; адреса повідомлення
            mov params.lpszCaption,offset szTitle1 ; адреса заголовка вікна
            mov params.dwStyle, MB_USERICON    ; стиль вікна
            mov params.lpszIcon, IDI_ICON2     ; ресурс позначки
            mov params.dwContextHelpId, 0      ; контекст довідки
mov params.lpfnMsgBoxCallback, 0 ;
            mov params.dwLanguageId, LANG_NEUTRAL  ; мова повідомлення
   invoke MessageBoxIndirect, ADDR params ;
  .ELSEIF ax==IDM_2 ; якщо вибрана кнопка меню "Требования к программе"
   invoke MessageBox,0,addr szInf2,addr szTitle2,MB_ICONWARNING+180000h ;
  .ELSEIF ax==IDM_3 ; якщо вибрана кнопка меню "Требования к интерфейсу окна"

   invoke MessageBox,0,addr szInf3,addr szTitle3,MB_ICONINFORMATION+180000h ;
  .ELSEIF ax==IDM_4 ; якщо вибрана кнопка меню "Элементы СП"
   invoke MessageBox,0,addr szInf4,addr szTitle4,MB_ICONINFORMATION ;
  .ELSEIF ax==Button1ID ;якщо повідомлення від кнопки закриття вікна "Delete"
    invoke PostQuitMessage,NULL
```

```
  .ELSEIF ax==Button4ID ; якщо повідомлення від кнопки прозорості "Расчеты"
      call PolPx4mod3
invoke DialogBoxParam, 0, IDM_POPUP, NULL, addr DialogProc, 0
    mov params.cbSize,SIZEOF MSGBOXPARAMS  ; розмір структури
              mov params.hwndOwner, 0           ; дескриптор вікна власника
   invoke GetModuleHandle, 0              ; отримання дескриптора програми
              mov params.hInstance, eax         ; збереження дескриптора програми
              mov params.lpszText, offset Msg2   ; адреса повідомлення
              mov params.lpszCaption,offset Titl2 ; адреса заголовка вікна
              mov params.dwStyle, MB_USERICON    ; стиль вікна
              mov params.lpszIcon, IDI_ICON3     ; ресурс іконки
              mov params.dwContextHelpId, 0      ; контекст довідки
              mov params.lpfnMsgBoxCallback, 0 ;
              mov params.dwLanguageId, LANG_NEUTRAL  ; мова повідомлення
   invoke MessageBoxIndirect, ADDR params ;
  .ENDIF
.ELSEIF uMsg==WM_LBUTTONDOWN ; якщо натиснута права кнопка миші
  invoke SendMessage,hWnd,WM_NCLBUTTONDOWN, HTCAPTION,0
.ELSE
    invoke DefWindowProc,hWnd,uMsg,wParam,lParam ; обробка та до WndProc
      ret             ; повернення з процедури
.ENDIF                ; закінчення логічної структури
  xor   eax,eax
  ret                                   ; повернення з процедури
WndProc endp                          ; закінчення процедури WndProc
```

PolPx4mod3 proc

```
.data
 fmt db "%d ",0  ; завдання перетворення одного символу
 szBuf1  db BSIZE dup(0),0    ; резервування пам'яті для буфера
 szBuf2  db BSIZE dup(0),0    ; резервування пам'яті для буфера
 szBuf3  db BSIZE dup(0),0    ; резервування пам'яті для буфера
 szBuf4  db BSIZE dup(0),0    ; резервування пам'яті для буфера
crtf    db 0dh,0ah
 fName   BYTE "PolPx4mod3.txt",0    ; комірки для імені файлу
 fHandle DWORD ?    ; резервування комірки для дескриптора збереження файлів
 cWritten  DWORD  ? ; резервування комірки для адреси символів виведення
a0 dd ?     ; коефіцієнт  при x0
a1 dd ?     ; коефіцієнт  при x1
a2 dd 0     ; коефіцієнт  при x2
a3 dd 0     ; коефіцієнт  при x3
a4 dd 0     ; коефіцієнт  при x4
;;;a5 dd ?     ; коефіцієнт  при x5
```

```
_h1_1 dd ?            ; початкове значення першої комірки
_h1_2 dd 0            ; початкове значення другої комірки
_h1_3 dd 0            ; початкове значення третьої комірки
_h1_4 dd 0            ; початкове значення четвертої комірки
;;;_h1_5 dd 0         ; початкове значення п'ятої комірки
 _m1 dd ?             ; робоче значення першої комірки
 _m2 dd ?             ; робоче значення другої комірки
 _m3 dd ?             ; робоче значення третьої комірки
 _m4 dd ?             ; робоче значення четвертої комірки
 ;;_m5 dd ?           ; робоче значення п'ятої комірки
_b dd 3
mm1 dd 0              ; лічильник рядків
meterLen dd 0 ; лічильник довжини циклу генерації
szSt1  db "%d%d%d%d%d   ",0
szBufS1 db "%d%d%d%d%d",0
szM db " %d   ",0
.code
invoke CreateFile, ADDR fName,  ; адреса імені файлу з символами
  GENERIC_WRITE,                                ; запис у файл
   0, NULL,                                     ; параметри багатозадачності
    CREATE_ALWAYS,              ; знищити та створити новий файл
     FILE_ATTRIBUTE_ARCHIVE, 0
  mov fHandle, eax          ; запам'ятовування дескриптора пристрою
  mov a4,1
  mov a0,1
  mov _h1_1,1
 @m1:
  m2m _m1,_h1_1 ;   1
  m2m _m2,_h1_2 ;   0
  m2m _m3,_h1_3 ;   0
  m2m _m4,_h1_4 ;   0
  ;;m2m _m5,_h1_5 ;   0
m1:    inc meterLen
mov edi,0

invoke wsprintf, ADDR szBufS1, ADDR szSt1,a4,a3,a2,a1,a0
invoke wsprintf, ADDR szBuf1[edi], ADDR fmt,_h1_1
invoke wsprintf, ADDR szBuf2[edi], ADDR fmt,_h1_2
invoke wsprintf, ADDR szBuf3[edi], ADDR fmt,_h1_3
invoke wsprintf, ADDR szBuf4[edi], ADDR fmt,_h1_4

mBegin:
 mov eax,_m1 ; підготування до перемноження
mul a1    ; виділення стану першої комірки
 mov ebx,eax ;
 mov eax,_m2 ;
```

```
mul a2 ; виділення стану другої комірки
 add eax,ebx ; отримання наступного стану першої комірки
  mov edx,0
  div _b
 mov ecx,edx

mov eax,_m3 ;
 mul a3 ; виділення стану третьої комірки
 add eax,ecx ;
  mov edx,0
  div _b
 mov ecx,edx

mov eax,_m4 ;
 mul a4 ; виділення стану четвертої комірки
 add eax,ecx ;
  mov edx,0
  div _b
 mov ecx,edx

 ;;mov eax,_m5 ;
 ;;mul a5 ; виділення стану п'ятої комірки
 ;;add eax,ecx ; отримання наступного стану першої комірки
 ;; mov edx,0
 ;; div _b
 ;;mov ecx,edx
 ; безпосередній зсув комірок
 ;;m2m _m5,_m4
   mov eax,_m3
   mul a3
   mov _m4,eax

   mov eax,_m2
   mul a2
   mov _m3,eax

   mov eax,_m1
   mul a1
   mov _m2,eax

mov _m1,ecx
; рахування кількості співпадань з першим станом матриці станів
   mov mm1,0
   mov eax,_m1 ; занесення робочого значення _m1
cmp eax,dword ptr _h1_1
    jnz m11
    inc mm1
```

```
m11: mov eax,_m2 ; занесення робочого значення _m2
    cmp eax,dword ptr _h1_2 ; порівняння з початковим значенням
    jnz m12
    inc mm1
m12: mov eax,_m3 ; занесення робочого значення _m3
     cmp eax,dword ptr _h1_3
    jnz m13
    inc mm1
m13: mov eax,_m4 ; занесення робочого значення _m4
    cmp eax,dword ptr _h1_4
    jnz m14
    inc mm1
m14:
    ;;mov eax,_m5 ; занесення робочого значення _m5
    ;;cmp eax,dword ptr _h1_5
    ;;jnz m15
    ;;inc mm1
;;m15:
  cmp dword ptr mm1,4  ; порівняння лічильника з 5
    jz m25
  inc edi
  inc edi
  inc meterLen
push edi
invoke wsprintf, ADDR szBuf1[edi], ADDR fmt,_m1
invoke wsprintf, ADDR szBuf2[edi], ADDR fmt,_m2
invoke wsprintf, ADDR szBuf3[edi], ADDR fmt,_m3
invoke wsprintf, ADDR szBuf4[edi], ADDR fmt,_m4
;;invoke wsprintf, ADDR szBuf5[edi], ADDR fmt,_m5
invoke wsprintf, ADDR szM, ADDR fmt,meterLen
 pop edi
jmp mBegin
m25:
invoke WriteFile, fHandle,   ; запис у файл за дескриптором пристрою
 ADDR szBufS1,       ; адреса області пам'яті, що зберігає символи
  BSIZE,                          ; кількість символів
   ADDR cWritten, NULL ; адреса пам'яті, де зберігається кількість
                                                                ; записаних у файл символів
invoke WriteFile,fHandle,ADDR crtf,2,ADDR cWritten,NULL

invoke WriteFile, fHandle,   ; запис у файл за дескриптором пристрою
 ADDR szBuf1,       ; адреса області пам'яті, що зберігає символи
  BSIZE,                          ; кількість символів
   ADDR cWritten, NULL       ; адреса пам'яті, де зберігається число
invoke WriteFile,fHandle,ADDR crtf,2,ADDR cWritten,0
invoke WriteFile, fHandle, ADDR szBuf2, BSIZE, ADDR cWritten, 0
invoke WriteFile,fHandle,ADDR crtf,2,ADDR cWritten,0
invoke WriteFile, fHandle, ADDR szBuf3, BSIZE, ADDR cWritten, 0
```

```
invoke WriteFile,fHandle,ADDR crtf,2,ADDR cWritten,0
invoke WriteFile, fHandle, ADDR szBuf4, BSIZE, ADDR cWritten, 0
invoke WriteFile,fHandle,ADDR crtf,2,ADDR cWritten,0
;;invoke WriteFile, fHandle, ADDR szBuf5, BSIZE, ADDR cWritten, 0
;;invoke WriteFile,fHandle,ADDR crtf,2,ADDR cWritten,0
invoke WriteFile,fHandle,ADDR crtf,2,ADDR cWritten,0
;invoke MemSet, ADDR szBuf1,'0',16
;invoke ZeroMemory,ADDR szBuf2,32
;invoke ZeroMemory,ADDR szBuf3,32
;invoke FlushFileBuffers, szBuf4
; очищення буферів
mov meterLen,0
 mov ecx,BSIZE
 mov esi,0
 mov al,0
@@: mov szBuf1[esi],al
 inc esi
 loop @b
   mov ecx,BSIZE
   mov esi,0
   mov al,0
@2: mov szBuf2[esi],al
   inc esi
   loop @2
mov ecx,BSIZE
mov esi,0
mov al,0
@3: mov szBuf3[esi],al
inc esi

loop @3
  mov ecx,BSIZE
  mov esi,0
  mov al,0
@4: mov szBuf4[esi],al
  inc esi
  loop @4
 mov ecx,BSIZE

add a1,1
.if a1<=2
 jmp m1
.endif
mov a1,0

 add a2,1
 .if a2<=2
  jmp m1
```

```
.endif
 mov a2,0

add a3,1
.if a3<=2
 jmp m1
.endif
mov a3,0

add a4,1
 .if a4<=2
  jmp m1
 .endif
 mov a4,1 ;;;0

add a0,1
.if a0<=2
mov _h1_1,2
 jmp @m1
.endif
   invoke  CloseHandle, fHandle                          ; закрити дескриптор файлу
ret
PolPx4mod3 endp

DialogProc proc hwndDlg: HWND, uMsg: UINT, wParam: WPARAM, \
     lParam: LPARAM
.Data

Screen     RECT <>  ; розміри екрана
 PopUp      RECT <>  ; розміри вікна, яке створюється
 TimerID    UINT ? ; идентификатор таймера
.code
switch uMsg
            case WM_INITDIALOG
invoke SystemParametersInfo, \ ; зміна настройок системи
  SPI_GETWORKAREA,\ ; визначає розмір робочого столу
    0 , addr Screen, 0
invoke GetClientRect, hwndDlg, \    ; витягує корд. робочої області вікна
   addr PopUp             ;  запис координат в структуру RECT з іменем PopUp
; обчислення відступу від правої межі екрана
  mov  eax, Screen.right   ; X-розмір екрана
  sub  eax, PopUp.right    ; відступ по X від правої межі екрана
  mov PopUp.left, eax                          ; X-розмір вікна
  m2m PopUp.top, Screen.bottom   ; Y-розмір вікна
  mov   ebx, PopUp.bottom            ; Рг для зміни координати Y
; цикл спливання вікна
    .repeat ; виконує блок, поки умова не істинна
```

```
  dec PopUp.top                    ; зменшення Y (для спливання вікна)
invoke MoveWindow, hwndDlg, \      ; змінює позицію і габарити вікна
   PopUp.left, PopUp.top, \        ; X та Y лівого верхнього кута
     PopUp.right, PopUp.bottom, \  ; ширина та висота вікна
       TRUE          ; приймає повідомлення WM_PAINT та перефарбовує
invoke ShowWindow, hwndDlg, SW_SHOW ; відобразити вікно на екрані
invoke UpdateWindow, hwndDlg ; оновлює вікно за допомогою WM_PAINT
invoke Sleep, 20         ; затримка на 20 мС
dec ebx
    .until zero?
mov TimerID, rv(SetTimer, hwndDlg, 0,1000,0)
            mov eax, TRUE
    case WM_TIMER
invoke KillTimer, NULL, TimerID
mov ebx, PopUp.bottom
 .repeat ; виконує блок, поки умова не істинна
inc PopUp.top         ; Y верхнього лівого кута
invoke MoveWindow, hwndDlg, \ ; змінює позицію і габарити вікна
  PopUp.left, PopUp.top, PopUp.right, PopUp.bottom, TRUE
invoke UpdateWindow, hwndDlg  ; оновлює вікно за допомогою WM_PAINT
invoke Sleep, 2           ; затримка на 2 мС
dec ebx
    .until zero?
    jmp _exit ; на примусове завершення
    case WM_CLOSE ; якщо є повідомлення про знищення діал. вікна
_exit:
invoke EndDialog, hwndDlg, 0 ; знищення діал. вікна
 default  ; switch-синтаксис
             xor eax, eax
    endsw
            mov eax, 0;TRUE
            ret
DialogProc endp
end start                        ; закінчення програми з ім'ям start
```

2 Файл ресурсів menu2.rc

```
#define IDM_1   1
#define IDM_2   2
#define IDM_3   3
#define IDI_ICON 22
#define IDI_ICON2 33
#define IDI_ICON3 44
IDI_ICON ICON DISCARDABLE MOVEABLE LOADONCALL "explorer2.ico"
IDI_ICON2 ICON DISCARDABLE MOVEABLE LOADONCALL  "user.ico"
IDI_ICON3 ICON DISCARDABLE MOVEABLE LOADONCALL  "messenger.ico"
```

```
FirstMenu MENU {
          POPUP "Автор"{
     MENUITEM "Автор программы",IDM_1}
          POPUP "Справка"{
     MENUITEM "Требования к программе",IDM_2
       MENUITEM SEPARATOR
     MENUITEM "Требования к интерфейсу окна",IDM_3}
          }
#define IDD_POPUP 37
IDD_POPUP DIALOGEX 10,10,250,70
CAPTION "Внимание!!!"
FONT 8,"MS Sans Serif",0,0,0
BEGIN
CTEXT "Расчет полиномов произведен.  Задержка на 1 секунду", 0, 40, 20, 180,
30
END
```

3 Командний bat-файл

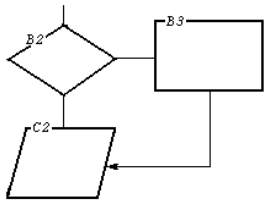
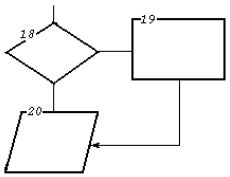
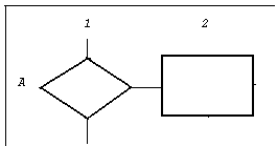
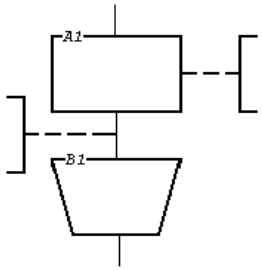
```
C:\masm32\bin\ml /c /coff "PolPx4mod3.asm"
C:\masm32\bin\rc "PolPx4mod3.rc"
C:\masm32\bin\link /SUBSYSTEM:windows "PolPx4mod3.obj" "PolPx4mod3.res"
pause
start PolPx4mod3.exe
```

Додаток Е

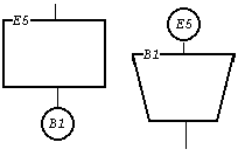
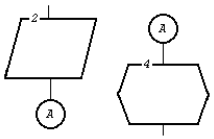
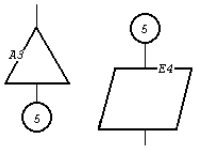
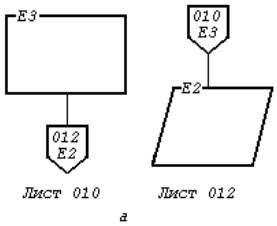
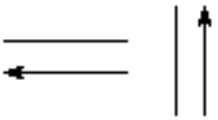
Витяг з ГОСТ 19.002-80 ЄСПД «Схемы алгоритмов и программ. Правила выполнения»

Застосування символів повинно відповідати вказаному в табл. Е.1

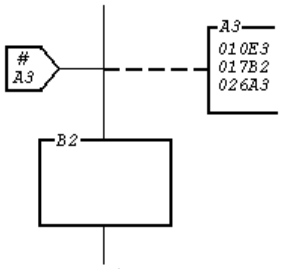
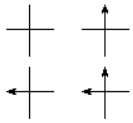
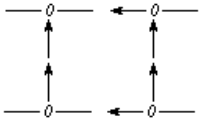
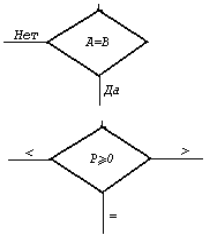
Таблиця Е.1

Фрагмент схеми	Зміст позначення	Правила застосування
1	2	3
	Можливі варіанти позначення символів у схемах: *B2, B3, C3* – координати зони аркуша, в якій розміщений символ	Координати зони символу або порядковий номер проставляють у верхній частині символу в розриві його контуру
	*18, 19, 20* – порядкові номери символів на схемі	
		Допускається не проставляти координати символів під час виконання схем від руки і за наявності координатної сітки
	Коментар	Застосовується, якщо пояснення не вміщається усередині символу (для пояснення характеру параметрів, особливостей процесу, ліній потоку і т.ін.). Коментар записують паралельно основному напису; розміщують у вільному місці схеми на даному аркуші і сполучають з пояснюваним символом

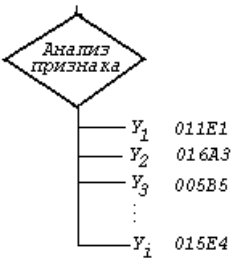
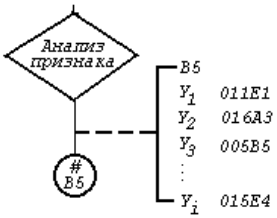
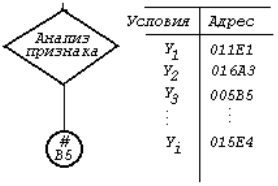
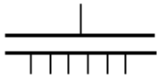
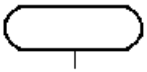
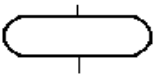
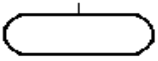
Продовження табл. Е.1

1	2	3
	З’єднувач: *E5, B1, A, 5* – ідентифікатори з’єднувача у вигляді: букви і цифри (координати зони аркуша)	Якщо схеми насичені символами, окремі лінії потоку між віддаленими один від одного символами допускається обривати. При цьому в кінці (початку) обриву повинен бути розміщений символ «З’єднувач»
	Букви	
	Цифри	
 *а*	Міжсторінковий з’єднувач. Перший рядок усередині міжсторінкового з’єднувача визначає номер аркуша, другий – координату символу	а) скріплені лінією потоку символи знаходяться на різних аркушах. *Примітка.* Під час виготовлення схем за допомогою ЕОМ допускається вказувати поряд з обривом лінії потоку адресні посилання без використання символів «З’єднувач» і «Міжсторінковий з’єднувач»
	Лінії потоку	Застосовують для вказівки напрямої лінії потоку: 1) можна без стрілки, якщо лінія спрямована зліва направо і зверху вниз; 2) із стрілкою – в інших випадках

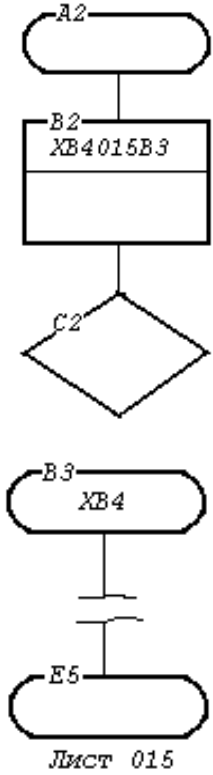
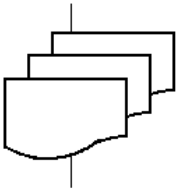
Продовження табл. Е.1

	*А3* – визначає зону на даному аркуші, де розташований символ «Коментар» *010Е3* – визначає номер аркуша і зону розташування, що пов'язуються з символом *Е3*	б) у разі зв'язку деякого символу з багатьма іншими символами, розташованими на різних аркушах, на вході цього символу поміщають один символ «Міжсторінковий з'єднувач», усередині якого на першому рядку розміщують знак # , а на другому рядку – координати символу «Коментар». Усередині символу «Коментар» указують номери сторінок і координати символів, пов'язаних з пояснюваним символом
	Злам лінії під кутом 90°	Позначає зміну напряму потоку
	Перетин ліній потоку	Використовується у випадку перетину двох незв'язаних потоків
	Злиття ліній потоку: місце злиття потоку позначене крапкою	Застосовується у випадку злиття ліній потоку, кожна з яких направлена до одного і того ж символу на схемі
	Місце злиття потоку позначене цифрою 0	Місце злиття ліній потоку допускається позначати точкою або цифрою 0
	Можливі варіанти відображення рішення: A = B, P ≥ 0 – умови рішень; A, B, P – параметри	Коли кількість результатів не більше трьох, ознака умови рішення (Так, Немає, =, <, >) проставляють над кожною лінією потоку, що виходить, або праворуч від лінії потоку

Продовження табл. Е.1

	y_i – умова i-го результату, *011E1, 016A3, 005B5, 015E4* – адреси результатів. Структура адреси має вигляд: XXX XX | |координата | символу |номер аркуша схеми	Коли число результатів більше трьох, умова результату проставляється в розриві лінії потоку. Адреса результату проставляється в продовженні умови результату і відділяється від нього пропуском
	*B5* – знак, який вказує, що умови рішення даються у вигляді таблиці або символу «Коментар», розташований на даному аркуші в зоні *B5*	У символі «З'єднувач» указують координату зони, куди мають поміщатися таблиця або символ «Коментар»
		У таблиці (у символі «Коментар») наводять адреси всіх переходів
	Паралельні дії: початок	Застосовується у разі одночасного виконання операцій, що відображаються декількома символами
	Початок, переривання і кінець алгоритму або програми: пуск	Символи застосовують на початку схеми алгоритму або програми, у разі переривання і в кінці. Усередині символу «Пуск – зупинка» може вказуватися найменування дії або ідентифікатор програми
	Переривання	
	Зупинка	

Закінчення табл. Е.1

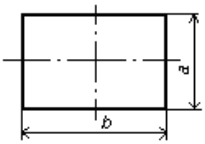
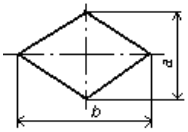
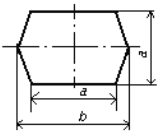
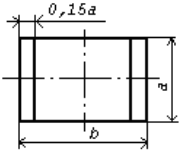
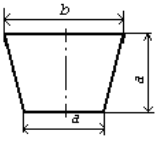
	Деталізація певної програми, подаваної в даній схемі одним символом: *1) XB4* – ідентифікатор програми; *2) 015* – номер аркуша, де проведений початок програми, що деталізується; *3) B3* – координата зони аркуша	Застосовується (на відміну від випадку, коли застосовується символ «Зумовлений процес») для деталізації у складі даної схеми програми. Програма, що деталізується, починається і закінчується символом «Пуск –зупинка». Усередині символу, за допомогою якого деталізує програма, проводять горизонтальну лінію. У даному прикладі програма, що деталізується, наведена за допомогою символу «Процес». Зліва над горизонтальною лінією поміщається ідентифікатор програми, що деталізується, а справа – номер аркуша і координата зони, де розміщений символ «Пуск – зупинка». Усередині символу «Пуск – зупинка», що позначає початок програми, яка деталізується, указується ідентифікатор даної програми
	Компактне подання безлічі носіїв даних однакового вигляду: документи	Застосовується, коли кожне з позначеної безлічі носіїв даних володіє певним набором властивостей і має лінії потоків одного вигляду і напряму

Додаток Ж

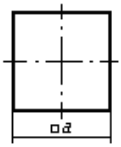
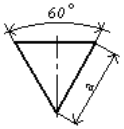
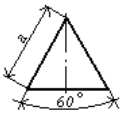
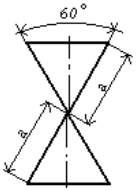
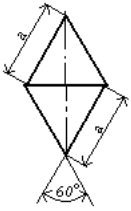
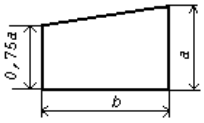
Витяг з ДСТУ 19.701-90 ЄСПД «Схемы алгоритмов и программ. Обозначения условные графические»

Перелік, найменування, позначення і розміри обов'язкових символів і функції, що відображаються ними, в алгоритмі і програмі обробки даних мають відповідати вказаним в табл. Ж.1.

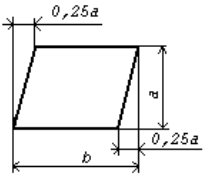
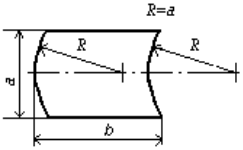
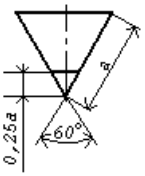
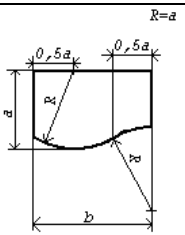
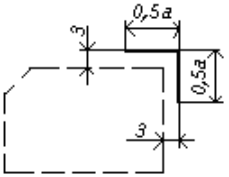
Таблиця Ж.1

Найменування	Позначення і розміри в мм	Функція
Процес		Виконання операцій або групи операцій, в результаті яких змінюється значення, форма уявлення або розташування даних
Рішення		Вибір напряму виконання алгоритму або програми залежно від певних змінних умов
Модифікація		Виконання операцій, які міняють команди або групу команд, що змінюють програму
Зумовлений процес		Використання раніше створених і окремо описаних алгоритмів або програм
Ручна операція		Автономний процес, що виконується вручну або за допомогою засобів, та діє неавтоматично
Лінія потоку		Вказівка послідовності між символами

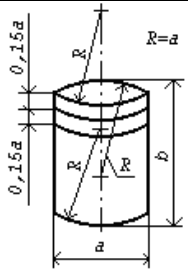
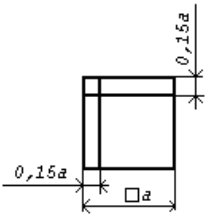
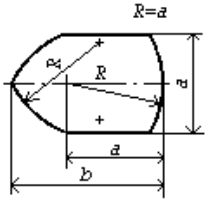
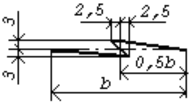
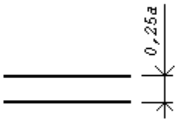
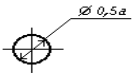
Продовження табл. Ж.1

Допоміжна операція		Автономний процес, що виконується пристроєм, не керованим безпосередньо процесором
Злиття		Об'єднання двох або більш множини в єдину множину
Виділення		Видалення однієї або декількох множин з єдиної множини
Групування		Об'єднання двох або більш множини з виділенням декількох інших множин
Сортування		Впорядкування множини за заданими ознаками
Ручне введення		Введення даних уручну за допомогою неавтономних пристроїв з клавіатурою, набором перемикачів, кнопок

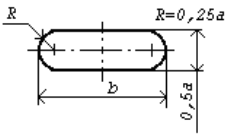
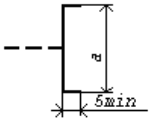
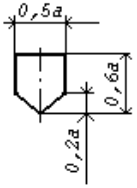
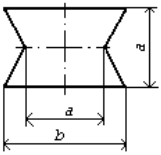
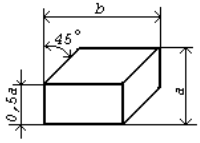
Продовження табл. Ж.1

Введення-виведення		Перетворення даних у форму, придатну для обробки (введення) або відображення результатів обробки (виведення)
Неавтономна пам’ять		Введення-виведення даних у разі використання пристрою, що запам’ятовує, керованого безпосередньо процесором
Автономна пам’ять		Введення-виведення даних у разі використання пристрою, що запам’ятовує, не керованого безпосередньо процесором
Документ		Введення-виведення даних, носієм яких служить папір
Файл		Уявлення організованих на основі загальних ознак даних, що характеризують в сукупності певний об’єкт обробки даних. Символ використовується у поєднанні з символами конкретних носіїв даних, що виконують функції введення-виведення

Продовження табл. Ж.1

Магнітний диск		Введення-виведення даних, носієм яких служить магнітний диск
Оперативна пам’ять		Введення-виведення даних, носієм яких служить магнітний сердечник
Дисплей		Введення-виведення даних, якщо безпосередньо підключений до процесу пристрій відтворює дані і дозволяє операторові ЕОМ вносити зміни в процесі їх обробки
Канал зв’язку		Передача даних каналами зв’язку
Паралельні дії		Початок або закінчення двох і більш операцій, які виконуються одночасно
З’єднувач		Вказівка зв’язку між перервними лініями потоку, зв’язуючими символами

Закінчення табл. Ж.1

Пуск – зупинка		Початок, кінець, переривання процесу обробки даних або виконання програми
Коментар		Зв'язок між елементом схеми і поясненням
Міжсторінковий з'єднувач		Вказівка зв'язку між роз'єднаними частинами схем алгоритмів і програм, розташованих на різних аркушах
Ручний документ		Формування документа в результаті виконання ручних операцій
Архів		Зберігання комплекту впорядкованих носіїв даних з метою повторного застосування

Розмір *a* необхідно обирати з ряду 10, 15, 20 мм. Допускається збільшувати розмір *a* на число, кратне 5. Розмір *b* дорівнює 1,5*a*.

Під час виконання умовних графічних позначень автоматизованим способом розміри геометричних елементів символів округляються до значень, визначених технічними можливостями використовуваних пристроїв.

Додаток К

Перелік Державних і міждержавних стандартів, що встановлюють вимоги до розробки програмних продуктів і програмної документації. Єдина система програмної документації

ГОСТ 19.001-77 ЕСПД. Общие положения.

ГОСТ 19.002-80 ЕСПД. Схемы алгоритмов и программ. Правила выполнения. (Заменен на ГОСТ 19.701-90).

ГОСТ 19.003-80 ЕСПД. Схемы алгоритмов и программ. Обозначения условные графические. (Заменен на ГОСТ 19.701-90).

ГОСТ 19.004-80 ЕСПД. Термины и определения. (Заменен на ГОСТ 19781-90).

ГОСТ 19.005-85 ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.

ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.

ГОСТ 19.102-77 ЕСПД. Стадии разработки.

ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.

ГОСТ 19.104-78 ЕСПД. Основные надписи.

ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.

ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.

ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.

ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению.

ГОСТ 19.301-79 ЕСПД. Программа и методика испытаний. Требования к содержанию и оформлению.

ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.

ГОСТ 19.402-78 ЕСПД. Описание программы.

ГОСТ 19.403-79 ЕСПД. Ведомость держателей подлинников.

ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.

ГОСТ 19.501-78 ЕСПД. Формуляр. Требования к содержанию и оформлению.

ГОСТ 19.502-78 ЕСПД. Общее описание. Требования к содержанию и оформлению.)

ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.

ГОСТ 19.504-79 ЕСПД. Руководство программиста. Требования к содержанию и оформлению.

ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению.

ГОСТ 19.506-79 ЕСПД. Описание языка. Требования к содержанию и оформлению.)

ГОСТ 19.507-79 ЕСПД. Ведомость эксплуатационных документов.

ГОСТ 19.508-79 ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению.

ГОСТ 19.601-78 ЕСПД. Общие правила дублирования, учета и хранения.

ГОСТ 19.602-78 ЕСПД. Правила дублирования, учета и хранения программных документов, выполненных печатным способом. (текст)

ГОСТ 19.603-78 ЕСПД. Общие правила внесения изменений.

ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом.

ГОСТ 19.701-90 (ИСО 5807-85) ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. (Взамен ГОСТ 19.002-80, ГОСТ 19.003-80).

Державні стандарти України

ДСТУ 2872-94 Система оброблення інформації. Мови програмування. Терміни та визначення.

ДСТУ 2873-94 Системи оброблення інформації. Програмування. Терміни та визначення.

ДСТУ 2874-94 Системи оброблення інформації. Бази даних. Терміни та визначення.

ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

СПИСОК ЛІТЕРАТУРИ

1. Кравець В. О. Системне програмування. Асемблер під Win32 API : навч. посіб. / В. О.Кравець, О. М. Рисований. – Х. : НТУ “ХПІ”, 2008. – 512 с.

2. Крупник А. Асемблер. Самоучитель / А. Крупник. – СПб. : Питер, 2005. – 235 с.: ил.

3. Магда Ю. С. Ассемблер для процессоров Intel Pentium / Ю. С. Магда. – СПб. : Питер, 2006. – 410 с.

4. Методичні вказівки до виконання та оформлення курсового проекту з курсу «Системне програмування» для студентів денної форми навчання за спеціальностями: 7.091501 «Комп’ютерні системи та мережі», 7.091502 «Системне програмування», 7.091503 «Спеціалізовані комп’ютерні системи» / уклад: О. М. Рисований. – Х. : НТУ «ХПІ», 2010. – 92 с.

5. Рисований О. М. Системне програмування [Текст] : підручник для студентів напряму “Комп’ютерна інженерія” вищих навчальних закладів / О. М. Рисований. – Х. : НТУ “ХПІ”, 2010. – 912 с.

6. Рисований О.М. Цифрові пристрої і мікропроцесори. Архітектура і програмне забезпечення : навч. посіб. / О. М. Рисований, М. В. Грушенко. – Х. : ХУПС, 2005. – 384 с.

ё

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання та оформлення курсового проекту
з курсу
Системне програмування
для студентів спеціальностей:
7.05010201 «Комп'ютерні системи та мережі»,
7.05010202 «Системне програмування»,
7.05010203 «Спеціалізовані комп'ютерні системи»

Укладач РИСОВАНИЙ Олександр Миколайович

Роботу до видання рекомендував *В. Д. Дмитрієнко*
Відповідальний за випуск *Ф. А. Домнін*

Редактор *Н. В. Верстюк*

План 2012 р., поз. № 176/

Підписано до друку . .2012. Формат 60x84 1/16. Папір офісний. Riso-друк.
Гарнітура Times. Ум. Друк арк.
Наклад прим.
Зам. № Ціна договірна.

Видавничий центр НТУ «ХПІ»
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ «ХПІ», 61002, Харків, вул. Фрунзе, 21