МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних та лабораторних робот

«Основи програмування мовою С++. Використання функцій»

з курсів
«Інформатика», «Основи інформаційних технологій»,
«Структури і алгоритми обробки даних»
для студентів спеціальностей: 152 «Метрологія та вимірювальна техніка»,
172 «Телекомунікація та радіотехніка» та 151 «Автоматизація та комп'ютерно-інтегровані технології»
денної та заочної форм навчання

Затверджено
редакційно-видавничою
радою університету,
протокол №1 від 19.02.2020

Харків
НТУ «ХПІ»
2020

Методичні вказівки до виконання практичних і лабораторних робіт «Основи програмування мовою С++. Використання функцій» з курсу «Інформатика», «Основи інформаційних технологій», «Структури і алгоритми обробки даних» для студентів спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології», 152 «Метрологія та вимірювальна техніка», 172 «Телекомунікація та радіотехніка» денної та заочної форм навчання / уклад. О. Є. Тверитникова, В.А. Крилова. – Харків : НТУ «ХПІ». – 28 с.

Укладачі О. Є. Тверитникова
В.А. Крилова

Рецензент д.т.н., професор Б.М. Горкунов

Кафедра інформаційно-вимірювальні технології і системи

ЗМІСТ

ВСТУП

Розвиток сучасних технологій неможливо без використання комп'ютерної техніки та програмного забезпечення. Підготовка фахівців в області автоматики, вимірювальної та медичної техніки вимагає великих знань і навичок володіння обчислювальною технікою, а так само знання основ алгоритмізації та програмування на мовах високого рівня таких як C/C ++.

Методичні вказівки призначені для вивчення мови програмування C++ на практичних заняттях та лабораторних роботах, а також для самостійного освоєння. У методичних вказівках детально і доступно розглянуті синтаксис, семантика і техніка програмування на мові C ++ з використанням функцій в C++. Описано всі етапи проектування програм, наведені докладні коментарі програмного коду, проаналізовані результати обчислень, показані типові проблеми та шляхи їх вирішення. Велика увага приділяється оголошення функції, тип функції, аргументи і параметри функції, передача аргументів по значенню та посиланню, локальні змінні, значення параметрів по замовчуванню, різні способи виклику функцій. Наведено індивідуальні завдання для виконання практичних та лабораторних робіт, також приведені і завдання для самостійного вивчення основ алгоритмізації та програмування на мові C++. Розглянуті приклади і завдання, допоможуть ефективному освоєнню основ програмування для учнів і початківців програмістів.

ЛАБОРАТОРНА РОБОТА

ВИКОРИСТАННЯ ФУКЦІЙ

Мета роботи – отримання практичних навичок написання та використання функцій в процесі розроблення програм мовою програмування С++.

Загальні теоретичні відомості

Принципи програмування на мові *С++* засновані на понятті функції. Використання функцій дозволяє досить ефективно побудувати процес розробки програмних проектів. Під час розроблення програми використовують стандартні вбудовані функції, які є складовою частиною мови програмування, наприклад, такі як *printf*(), s*canf*(), математичні функції бібліотеки *math* (*fabs*(), *s*in(), ...) (Додаток А).

Головною частиною програми також є функція – це функція *main*(). Виконання програми завжди починається з команд, що містяться в функції *main*(), а всередині її можуть викликатися інші функції. Крім того існують функції, які створюються користувачем для власних потреб.

Функція – це самостійна одиниця програми, спроектована для реалізації конкретного завдання. Кожна функція має власне ім’я, за яким здійснюють її виклик. Виклик функцій призводить до виконання деяких дій.

Створювана у програмі функція повинна бути оголошеною і визначеною. Оголошення функції має бути написаним у програмі раніш за її використання. Визначення може перебувати у будь-якому місці програми, за винятком тіла (середини) інших функцій.

Оголошення функції складається з заголовка і має форму:

тип ідентифікатор (формальні параметри)

Реалізація функції, крім заголовка, містить тіло функції (команди, які виконує функція) і команду повернення результату *return*.

Реалізація функції в програмі має наступну структуру:

тип ідентифікатор (формальні параметри)

```
{
  опис 1;
  опис 2;
  ...
  оператор 1;
  оператор 2;
  return вираз;
}
```

Параметр «ідентифікатор» функції це символьне ім'я, по якому виконується виклик даної функції. Параметр «тип» функції визначає тип значення, яке повертає дана функція. В процесі виконання функція може обчислювати деяке значення і повертати його викликає функції. Наприклад, функція, яка обчислює суму деякої послідовності чисел, може повертати значення цієї суми. Також, значення, які повертаються функціями, використовуються для діагностики правильності виконання функції – якщо функція виконана успішно, то функція повертає деяке значення, якщо в процесі виконання функції виникла помилка, то функція може повернути її код. Тип функції може бути стандартним або призначеним для користувача. Крім того, функція може не повертати значення. У цьому випадку тип функції вказується як *void*.

Виконання функції завершується при виконанні оператора повернення *return*. Параметром оператора є деякий вираз, значення якого повертає функція. Функція може містити кілька операторів повернення. Для функцій типу *void* параметр вираз в операторі повернення відсутня, і оператор повернення може бути опущений. У цьому випадку вихід з функції виконується після завершення виконання останнього оператора тіла функції.

Виклик функції. Для виклику функції в програмі необхідно вказати її ім'я і задати перелік необхідних параметрів виклику.

Наприклад: реалізація функції визначення мінімального значення з двох аргументів.

```
#include <iostream>
float min (float fst, float sec)
{
  if(fst <sec)
```

```
        return fst;
  else
        return sec;
}
void main ()
{
  float val1, val2;
  cin >> val1 >> val2;
  cout << "мінімальне значення" << min (val1, val2) << '\ n';
}
```

Функція *min* (*a, b*) викликається в функції *main* () як частина оператора виведення. При виконанні цього оператора відбуватися виклик функції *min*, яка обчислює деяке значення і це значення підставляється в оператор виведення і виводиться на екран. Функція *min* містить два параметри. При виконанні функції значення змінної val1 копіюється в формальний параметр *fst*, а значення змінної *val2* копіюється в параметр *sec*. При виконанні функції формальні параметри порівнюються. Функція повертає значення меншого з формальних параметрів.

Раніше було відзначено, що функція повинна бути описана до її першого використання. Тому, у всіх наведених прикладах спочатку розглядалася реалізація функції, а потім її виклик. У деяких випадках неможливо дотримати таку структуру програми. Тоді використовують прототипи функцій.

Прототипом функції в мові *C++* називається оголошення функції, яке не містить тіло функції, але вказує ім'я функції, кількість і типи формальних параметрів, і повертається функцією тип даних:

```
void func (int arg1, float arg2);// прототип функції
void main()
{
  ...
  float x;
  func(20, x);                    // виклик функції
  ...
}
void func(int arg1, float arg2)  / /реалізація функції
{
  // тіло функції
}
```

Вбудовані та перевантажені функції. У мові *C++* є можливість використовувати однакові імена для різних функцій. Така необхідність виникає, коли, наприклад, потрібно виконувати однакові дії над змінними різних типів. Таку можливість дає механізм перевантаження функцій. У *C++* двом або більше функцій може бути дано одне й те саме ім'я за умови, що їх списки параметрів розрізняються або числом параметрів, або їх типами. Вибір конкретної реалізації залежить від типу аргументів функції.

Наприклад:

int max(int, int); double max(double, double)

Вбудовані функції в *C++* визначаються за допомогою ключового слова *inline*. Воно розташовується перед оголошенням функції, коли необхідно, щоб код в тілі функції вбудовувався в місце виклику функції.

*inline double cube (double x) { return (x*x*x); }*

Обмеження компілятора не дозволяють вбудовувати складні функції. Крім того, ключове слово *inline* є лише рекомендацією компілятору.

Рекурсія. Рекурсією називається конструкція, коли деяка функція викликає саму себе.

Наприклад:

```
int sum (int n)
{ if (n == 1)
 return 1;
 else
return n + sum(n-1);
}
```

Також прикладом рекурсивного вираження є обчислення факторіала:

$n! = 1 * 2 * ... (n–1) * n$

З іншого боку можна помітити, що

$(N–1)! = 1 * 2 * ... * (n –1)$

і, отже, обчислення $n!$ можна поставати як

$n! = (N–1)! * n.$

Це є рекурентна формула для обчислення факторіала деякого цілого числа. Тобто можна написати наступну функцію.

```
int factorial (int n)
{
 if (N> 1)
       return factorial (N–1);
 else
       return 1;
}
```

Тут для обчислення значення функції для параметра n викликається ця ж функція для параметра (*n*–1). Але рекурсивна програма не може викликати себе нескінченно, інакше вона ніколи не зупиниться, таким чином в програмі (функції) повинна бути присутнім ще один важливий елемент – так зване термінальне умова, тобто умова при якому програма припиняє рекурсивний процес. У прикладі такою умовою є повертається функцій значення при *n* == 1.

Передача параметрів в функцію. Передача значення виконується з використанням списку формальних параметрів. Значення і змінні, які вказуються в дужках при виконанні функції, називаються фактичними параметрами.

```
тип функція (тип формальний_параметр1, тип формальний_параметр 2, тип формальний_параметр 3, ...) // опис функціїї
{
……..
}
...
функція (фактичний_параметр1, фактичний_параметр2, фактичний_параметр3, ...);          / /виклик функції
```

Формальні параметри в заголовку функції задаються списком. У списку кожне значення є ідентифікатор формального параметра із зазначенням його типу. Один від одного параметри в списку відділені комами. При виконанні функції фактичні параметри також представлені у вигляді списку елементів, розділених комами. Список фактичних параметрів повинен відповідати списку формальних параметрів.

При виконанні функції це відповідність дотримується наступним чином – значення першого фактичного параметра присвоюється першому формальному параметру, значення другого фактичного параметра присвоюється другому формальному параметру тощо (рис. 1)

Передача значень може бути виконана кількома способами:

- передача параметра за значенням;
- передача параметра за вказівником;
- передача параметра за посиланням.

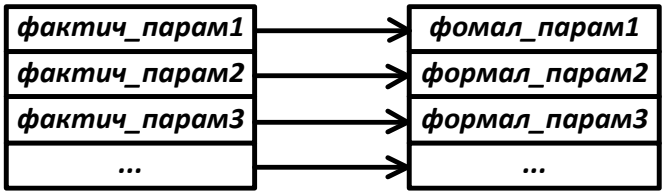

Рисунок 1 – Передача в функцію фактичних параметрів

Передача параметра за значенням. Даний метод є найпоширенішим і використовується, коли необхідно передати деяке значення в функцію для використання всередині функції. В цьому випадку формальні параметри є локальними змінними всередині функції. При виконанні функції значення фактичних параметрів копіюються у відповідні їм формальні параметри.

У наступному прикладі функція використовується для визначення потрапляє деяке значення в заданий діапазон.

```
bool in_range (float val, float low, float hi)
{
  if ((val> low) && (val <hi))
        return true;
  else
        return false;
}
void main ()
{
  float x, min, max;
  ...
```

```
 if (in_range (x, min, max))
       cout << "Потрапляє" << '\ n';
 ...
 if (in_range (x, 0.f, 1.f))
       cout << "Знову потрапляє" << '\ n';
 ...
}
```

Функція *in_range* має три формальних параметра. Перший параметр *val* містить значення, яке перевіряється на потрапляння в діапазон. Параметри *low* і *hi* містять значення нижньої і верхньої меж цього діапазону. При виконанні функції змінна *val* порівнюється з межами діапазону. Якщо її значення більше нижньої межі і менше верхньої, то функція поверне значення *true*, інакше функція поверне значення *false*.

При першому виклику функції в прикладі фактичні параметри є змінними. Значення змінної x присвоюється формальному параметру *val*, значення змінної *min* присвоюється формальному параметру *low*, а значення змінної *max* присвоюється формальному параметру *hi*. І в даному випадку функція *in_range* визначить лежить чи ні значення x в діапазоні значень *min .. max*.

При другому виконанні функції показано, що в якості фактичних параметрів можна вказувати не тільки змінні, але і константи. У цьому прикладі формальному параметру *low* буде присвоєно значення 0.*f*, а формальному параметру *hi* буде присвоєно значення 1.*f*. Функція визначить чи лежить значення *x* в діапазоні значень 0..1.

Можна також запропонувати більш короткий спосіб реалізації функції *in_range*.

```
bool in_range (float val, float low, float hi)
{
 return ((val> low) && (val <hi));
}
```

Таке можливо, якщо згадати якого типу результат повертають операції відносини і логічні операції. При виконанні функції буде обчислено значення виразу в операторі повернення. Результат цього виразу має логічний тип зі значеннями *true* або *false*.

Передача параметра за вказівником. При використанні передачі в функцію параметра за значенням, є можливість передати тільки значення в функцію, але не можна отримати значення з функції.

Наприклад: реалізація функції, яка обмінює значення двох цілих змінних.

```
void swap (int u, int v)
{
  int t = u;
  u = v;
  v = t;
}

void main ()
{
  int x = 10, y = 124;
  ...
  cout << "До x = "<< x << " <<" y = "<< y << '\ n';
  swap (x, y);
  cout << "Після x = "<< x << " <<" y = "<< y << '\ n';
  ...
}
```

При виконанні висновок програми буде таким

до x= 10 y = 124
після x= 10 y = 124

Таким чином, незважаючи на те, що змінилися значення формальних параметрів всередині функції, відповідні їм значення фактичних параметрів залишилися незмінними.

Для передачі в зухвалу функцію змінених значень параметрів використовується передача параметра за вказівником. При використанні цього методу формальний параметр повинен бути заданий як покажчик. Відповідний йому фактичний параметр при цьому повинен задавати адресу деякої змінної. При виконанні функції в неї передається не значення фактичного параметра, а адреса де зберігається змінна фактичного параметра. Всі маніпуляції всередині функції виконуються із значенням,

яке зберігається за поданою адресою і, отже, всі зміни, які зробила функція з цим значенням, збережуться після її завершення.

Наведений вище приклад повинен бути реалізований так

```
void swap (int * u, int * v)
{
  int t = * u;
  * U = * v;
  * V = t;
}
void main ()
{
  int x = 10, y = 124;
  ...
  cout << "До x = "<< x << " <<" y = "<< y << '\ n';
  swap ( & X, & y);
  cout << "Після x = "<< x << " <<" y = "<< y << '\ n';
  ...
}
```

Крім того в функції swap описана змінна t. Така змінна називається локальної змінної, і використовувати її можна тільки в цій функції.

Передача параметра за посиланням. Ще один спосіб передачі значень з деякої функції в зухвалу функцію є передача параметра за посиланням. Посилання – це альтернативне ім'я деякої змінної. Формальний параметр може бути описаний як посилання наступним чином

```
тип & параметр
```

Можна розглянути використання посилань на наступному прикладі. Реалізована нижче функція змінює значення деякої дійсної змінної на її абсолютне значення.

```
void to_abs (double & arg)
{
  if (arg <0)
        arg = –arg;
}
```

Виклик функції виконується так

```
double double_val;
...
to_abs (double_val);
```

У зухвалій функції оголошена змінна *double_val*. Формальний параметр *arg*, оголошений як посилання є, по суті, ще одним ім'ям змінної *double_val*. І це альтернативне ім'я використовується всередині функції для обчислень. Таким чином всі зміни, які будуть виконуватися в функції зі змінною *arg* будуть виконані зі змінною *double_val*.

Використання посилань для передачі параметрів у функцію ще цікаво тим, що при такій передачі не виконується копіювання даних з фактичного параметра в формальний. При передачі у функцію змінної великого розміру за значенням таке копіювання може займати значний час і в деяких випадках доцільно замінити передачу параметра за значенням на передачу параметра за посиланням. Крім того, при передачі параметра за значенням фактичний параметр не змінюється після завершення функції. Якщо Ви використовуєте дзвінок параметра за посиланням як альтернатива передачі за значенням і передбачається, що параметр не буде змінений всередині функції можна заборонити можливість такої зміни. Для цього використовуються константні посилання. У прикладі функція обчислює довжину вектора в тривимірній системі координат по його відомим координатам.

```
#include <math.h>
void vec3d_len (const double & x, const double & y, const double & z, double & len)
{
  len = sqrt(pow (x, 2) + pow (y, 2) + pow (z, 2));
}
```

Функція буде викликатися так:

```
double coord1, coord2, coord3, vector;
...
vec3d_len (coord1, coord2, coord3, vector);
```

У функції *vec3d_len* перші три формальних параметра є константними посиланнями і їх не можна змінити всередині функції. Четвертий параметр описаний як звичайне посилання і його значення обчислюється всередині функції. Виклик функції виконується без використання покажчиків.

Отже, при організації функції потрібно дотримуватися таких основних правил:

1. Кількість фактичних параметрів функції має збігатися з кількістю її формальних параметрів.
2. Типи змінних, які є фактичними параметрами, повинні збігатися з типами відповідних формальних параметрів.
3. Імена змінних, які є фактичними параметрами, не обов'язково співпадають з іменами формальних параметрів.
4. Фактичними параметрами можуть бути константи, змінні чи вирази.
5. У прототипі функції необов'язково задавати імена формальних параметрів, достатньо їх оголосити, наприклад у такий спосіб:
6. Якщо функція не отримує жодного параметра, слід у заголовку функції ставити порожні дужки (опустити дужки не можна) або записати у дужках *void*.
7. Тип значення, яке повертає функція, може бути яким завгодно, але не може бути масивом чи функцією.
8. *C++* дозволяє існування у програмі функцій з однаковим ім'ям, але різними параметрами.

ПРИКЛАДИ ВИРІШЕННЯ ЗАВДАННЬ

***Завдання* 1.**

Обчислити значення виразу $y=\frac{\sqrt{7}+7}{2}+\frac{\sqrt{15}+15}{2}+\frac{\sqrt{25}+25}{2}$.

Складовими даного виразу є схожі формули, кожну з яких можна записати у вигляді – $\frac{\sqrt{x}+x}{2}$, де x=7, або x =15, або x =25. Тобто, необхідно створити функцію обчислення, параметром якої буде x.

Синтаксис програми:

```
#include <iostream>
#include <math.h>
using namespace std;
double function (double x) // реалізація функції
{
return (sqrt(x)+x)/2;
}
int main ()
{
double y = function (7) + function (15) + function (25); // виклик функції
cout<<"Результат y = "<< y <<'\n'; // вивод на косоль результату
   return 0;
}
```

***Завдання* 2.**

Обчислити значення виразу $\frac{\sqrt{7}+21}{7+\sqrt{21}}+\frac{\sqrt{5}+34}{5+\sqrt{34}}+\frac{\sqrt{56}+6}{56+\sqrt{6}}$

У даному виразі функція має два параметри.

Синтаксис програми:

```
include <iostream>
#include <math.h>
using namespace std;
double function (double x, double z) // реалізація функції
{
return (x+sqrt(z))/(z+sqrt(x));
}
int main ()
{
double x = function (7,21) + function (5,34) + function (56,6); //виклик функції
cout<<" Результат y = "<< y << '\n'; // вивод на косоль результату
return 0;
}
```

***Завдання* 3**.

Створити функцію обчислення найбільшого спільного дільника (НСД) двох натуральних чисел та визначити за її допомогою НСД для двох і трьох введених чисел.

Обчислення НСД можна здійснити різними способами.

1. Розкладення чисел на прості множники полягає у простому перебиранні всіх чисел з перевіркою на подільність без залишку обох чисел. Перше число, для якого виконається умова перевірки, і буде НСД. Недоліком цього способу є мала швидкість роботи.

```
#include <iostream>
using namespace std;

int nod1( int a,   int b)
{
   int n=1;
   if(a > b) n = b; else n = a;
   n++;
   do  n--;
   while( a%n != 0 || b%n != 0);
   return n;
}
```

2. Існує декілька алгоритмів Евкліда обчислення НСД пари додатних цілих чисел, які використовують різні рекурентні співвідношення. Бінарний алгоритм Евкліда більш швидкий, ніж звичайний алгоритм Евкліда, оскільки замість повільних операцій віднімання виконуються зсуви. Бінарний алгоритм Евкліда – добре відома процедура знаходження найбільшого спільного дільника (НСД) двох цілих чисел. В основі цього алгоритму лежить наступне твердження. Нехай a і b два цілих числа або многочлена, причому $a \geq b$ або $deg\ [(a)] \geq deg\ [(b)]$. Розділимо a на b. Якщо залишок від ділення r дорівнює 0, то $d = b$ є найбільшим спільним дільником. Якщо залишок не дорівнює 0, замінимо a на b, b на r і повторимо розподіл.

Наприклад знайти НСД цілих чисел $a = 186$ і $b = 66$. Дотримуючись алгоритму Евкліда отримуємо

Крок 1. $r = 186 \ mod \ 66 = 54$, $r \neq 0 \rightarrow a = 66$, $b = 54$.
Крок 2. $r = 66 \ mod \ 54 = 12$, $r \neq 0 \rightarrow a = 54$, $b = 12$.
Крок 3. $r = 54 \ mod \ 12 = 6$, $r \neq 0 \rightarrow a = 12$, $b = 6$.
Крок 4. $r = 12 \ mod \ 6 = 0$, $r = 0$.
Таким чином, НСД (186, 66) = 6.

Текст функції знаходження НСД двох цілих чисел та основної програми:

```
int nod2( int a,   int b)
{
   while ( a && b )
      if ( a>=b ) a %= b;
      else      b %= a;
   return a | b;
}

int main()
{
 int x, y;
 cout <<" Введіть два натуральних (цілих додатних) числа \n --> ";
 cin >> x;>> y;
 if (x<=0 || y<=0) cout << "Некоректні дані!" << endl;
         else
         {
         cout<<"НСД:  "<< nod1(x,y)<<endl;
         cout<<"НСД:  "<< nod2(x,y) <<endl;
         }
 system ("pause>>void");
 return 0;
 }
```

Порядок виконання лабораторної роботи

1. Відповідно до номером по журналу виберіть індивідуальне завдання.

2. Розробіть алгоритм вирішення завдання.

3. Складіть текст програми.

4. Створіть проект в інтегрованому середовищі розробки Microsoft *Visual Studio* (*lab_function_*призвище.*cpp*).

5. Введіть текст програми.

6. Скомпілюйте програму. Якщо в програмі є помилки, їх необхідно виправити. Якщо помилок немає, то з'явиться повідомлення про успішну компіляції.

7. Запустіть програму на виконання, проаналізуйте результати і переконайтеся в правильності рішення задачі.

8. Виконайте звіт з лабораторної роботи, який повинен містити: титульний аркуш; мета роботи; індивідуальне завдання; алгоритм роботи програми; текст програми; результати роботи програми; висновки.

Зміст звіту

1. Мета лабораторної роботи
2. Завдання лабораторної роботи
3. Короткі теоретичні відомості
4. Текст програми
5. Результати роботи програми
6. Висновки

Завдання для індивідуальної роботи

1. Створити функція обчислення $\frac{x+sin(y)}{y+sin(x)}$ та визначити $\frac{1+sin(5)}{5+sin(1)}+\frac{3+sin(7)}{7+sin(3)}+\frac{4+sin(14)}{14+sin(4)}$ за допомоги цієї функції.

2. Обчислити для двох введених чисел *a* та *b* значення

а. $z=\min(a,2b)+\min(2a-b,b)$, створивши функцію визначення найменшого з двох чисел.

3. Створити функцію обчислення гіпотенузи прямокутного трикутника за значеннями двох катетів та за допомоги цієї функції визначити для двох введених значень катетів значення гіпотенузи/

4. Створити функцію визначення кількості цифр числа та за її допомоги обчислити розмірності трьох введених чисел.

5. Створити функцію обчислення найбільшого спільного дільника (НСД) двох натуральних чисел за алгоритмом Евкліда та за допомогою неї визначити НСД чотирьох введених чисел.

6. Дано натуральне число n. Обчислити 1!+2!+3!+.....+n! Визначити функцію обчислення факторіала числа.

7. Написати функцію, яка перевіряє, чи є число паліндромом.

8. Знайти всі прості числа із заданого інтервалу. Використайте функцію, яка визначає, чи є число простим.

9. Написати функцію, яка перевіряє, чи є число досконалим, тобто чи дорівнює воно сумі своїх дільників, за винятком самого себе.

10. Написати функцію знаходження суми цифр числа.

11. Написати програму пошуку найбільшого з трьох чисел, використовуючи функцію пошуку максимального з двох чисел.

12. Реалізувати рекурсивну функцію для обчислення найбільшого спільного дільника двох натуральних чисел A та B, використовуючи алгоритм Евкліда.

13. Створити функцію обчислення факторіала та визначити: $z = \frac{3*7!-2*6!}{5!+3!}$

14. Створити функцію визначення абсолютного значення (за модулем) числа без використання стандартних математичних функцій) та обчислити за її допомоги модуль значення для трьох введених чисел.

15. Створити функцію обчислення середнього геометричного чотирьох чисел за формулою $\sqrt[4]{a*b*c*d}$ та за допомоги цієї функції визначити для чотирьох введених чисел значення середнього геометричного.

Завдання для самостійної роботи 1

1. Створити одновимірний масив, що складається з подільників десяткового цілого числа. Створити функцію, яка визначає місцеположення максимального елемента одновимірного масиву.

2. Створити матрицю розміром 5x5 за допомогою генератора цілих чисел, вивести її на екран. Знайти суму елементів в тих рядках, які містять тільки позитивні числа. Створити функцію перетворення матриці в одновимірний масив і знаходження максимального елемента в одновимірному масиві.

3. Побудувати двовимірний масив дійсних чисел. Визначити місце розташування двох найбільших чисел. Створити функцію обміну максимального елемента головної діагоналі і мінімального елемента побічної діагоналі матриці.

4. Створити двовимірний масив розміром 10x15. Знайти суму елементів кожного рядка двовимірного масиву і записати ці суми в одновимірний масив. Створити функцію знаходження максимального елемента отриманого масиву

5. Ввести з клавіатури число *N*. Створити масив, що складається з цифр даного числа. Створити функцію знаходження мінімального елемента створеного одновимірного масиву, перевірити чи є отримане число простим.

6. Створити двовимірний масив розміром 10x15. Створити одновимірний масив, що складається з максимальних елементів стовпців двовимірного масиву. Створити функцію сортування отриманого одновимірного масиву по зростанню.

7. Створити двовимірний масив 10x10. Поміняти місцями максимальний і мінімальний елементи масиву. Створити функцію, знаходження суму елементів головної діагоналі масиву.

8. Дана прямокутна матриця. Поміняти місцями рядок, що містить елемент з максимальним значенням, з рядком, що містить елемент з найменшим значенням. Створити функцію сортування першого стовпчика по спадаючій.

9. Дан двовимірний масив цілих чисел. Поміняти місцями максимальний елемент масиву і мінімальний елемент його головної

діагоналі. Створити функцію знаходження суми елементів другого рядка двовимірного масиву.

10. Для заданого числа *N* побудувати одновимірний масив, що складається з *N* чисел Фібоначчі. Створити функцію, що визначає значення третього парного числа.

11. Для заданого числа *N* отримати всі скоєні числа менше *N*. Створити функцію знаходження суми отриманих чисел.

12. Створити одновимірний масив, що складається з подільників цілого числа *n*, що вводиться з клавіатури. Створити функцію, яка перетворює отриманий масив в зворотний.

13. Побудувати двовимірний масив дійсних чисел. Визначити місце розташування максимального і мінімального чисел. Створити функцію знаходження суму елементів побічної діагоналі.

14. У двовимірному масиві визначити добуток суми індексів двох максимальних елементів не головною діагоналі. Створити функцію сортування останнього стовпця масиву по спадаючій.

15. У двовимірному масиві знайти суму максимальних значень елементів її рядків. Створити функцію перетворення двовимірного масиву в одновимірний.

Завдання для самостійної роботи 2

Для цілого *n* і дійсних *R, x* і *y* визначити значення *REZULT*. Реалізувати дві рекурсивні функції:

- $\Phi(n)$ – *n*-е число Фібоначчі
  $a_a=a_2=1$, $a_3=2$, $a_4=3$, $a_5=5$, $a_6=8$ тобто $a_n= a_{n-1} + a_{n-2}$.
- $n!$ – факторіал *n*.

$$1.\ \text{REZULT} = \begin{cases} \dfrac{\sin^3\left(\dfrac{x}{y}\right)\cdot\left(2(x+y^2)-\sqrt{25.8x-134}\right)}{\left(\dfrac{x}{y}\cdot 12.002-\sin(R-2.002)\right)}+\Phi(n),\ \textit{при}\ \ n\ge 1,\ \ x>0\ \ \textit{та}\ \ y>0 \\ \dfrac{\cos^2(x+y)+\sin(x^2+y^2)}{\sqrt{xy/tg(x^2+y^2)}}+n!,\ \ \textit{при}\ \ n\ge 1,\ \ x<0\ \ \textit{та}\ \ y>0 \\ 12\ -\ \textit{в останніх випадках} \end{cases}$$

$$2.\ \mathrm{REZULT} = \begin{cases} \cos^3\left(\frac{x}{y}\right) + \frac{5^{3x^2}}{R} + \Phi(n),\ при\ n \ge 1,\ x > 1\ та\ y > 5 \\ \frac{ctg^5(x+y) + tg^5(x+y)}{\sqrt{5.2x^{10} - 1.5x^{22} - 4.5}} + n!,\ при\ n \ge 1,\ x < 0\ та\ y < -1 \\ 10 - в\ останніх\ випадках \end{cases}$$

$$3.\ \mathrm{REZULT} = \begin{cases} \frac{3x^2 + 4 \diagup x^2(x^2+1)^3}{\sin x \cos x + (\cos 2x - 3)} + n!,\ при\ n \ge 1,\ x < 0\ та\ y < 0 \\ \frac{1}{\sin\frac{x}{2}\sqrt{\cos^3\frac{x}{2} + 34}} + \Phi(n),\ при\ n \ge 1,\ x > 0\ та\ y > 2 \\ R - в\ останніх\ випадках \end{cases}$$

$$4.\ \mathrm{REZULT} = \begin{cases} \frac{\sin^2 x}{1 - tgx} + \frac{1}{4 - 3\cos^2 x + 5\sin^2 x} + \Phi(n),\ при\ n \ge 1,\ x < 0\ та\ y < 0 \\ \frac{x^3 - 6x^2 + 11x - 5 - R1}{(x-2)^4} + \frac{\sin^3 x}{\cos^4 x} + n!,\ при\ n \ge 1,\ x < 0\ та\ y < 0 \\ R - в\ останніх\ випадках \end{cases}$$

$$5.\ \mathrm{REZULT} = \begin{cases} \sqrt{\frac{\cos^2(\ln(\frac{x}{y})) + n!}{\sqrt{5x^{10}} + 246x^8}} + \Phi(n),\ при\ n \ge 1,\ x \ge 0\ та\ y < 0 \\ \frac{\sin^3(x-y) - 31 + n!}{x\sqrt{1 - (\ln x)^2}} R,\ при\ n \ge 1,\ x \ge 1\ та\ y \ge 2 \\ R - в\ останніх\ випадках \end{cases}$$

$$6.\ \text{REZULT} = \begin{cases} \dfrac{1/\cos x \sin^3 x}{\cos^4 x\sqrt{4-ctg^2 x}} \cdot R + \Phi(n),\ при\ n \ge 1,\ x < 0\ та\ y < 0 \\ \sqrt{\dfrac{1-\sqrt{x}}{1+\sqrt{x}}} + \dfrac{tgx}{1+tgx+tg^2 x} + n!,\ при\ n \ge 1,\ x < 0\ та\ y < 0 \\ R - в\ останніх\ випадках \end{cases}$$

$$7.\ \text{REZULT} = \begin{cases} \dfrac{\sin^2\left(\dfrac{y}{2\cdot x}\right)\cdot\left(2(x)\sqrt{34\cdot y+64.4}\right)}{\left(\dfrac{(x+y)}{y}-\sin(x)\right)} + \Phi(n),\ при\ n \ge 1,\ x < -5\ та\ y > 0 \\ \dfrac{\cos^2(x+y)+\sin(x^2+y^2)}{\sqrt{xy/tg(x^2+y^2)}} + n!,\ при\ n \ge 1,\ x < 0\ та\ y > 0 \\ 15 - в\ останніх\ випадках \end{cases}$$

$$8.\ \text{REZULT} = \begin{cases} \sqrt{\dfrac{\cos^2(\dfrac{x}{y})+n!}{\sqrt{5x^{10}}+y}} + \Phi(n),\ при\ n \ge 1,\ x \ge 0\ та\ y < 0 \\ \dfrac{\sin^3(x-y)n!}{x\sqrt{1-(\ln x)^2}} R,\ при\ n \ge 1,\ x \ge 1\ та\ y \ge 2 \\ R - в\ останніх\ випадках \end{cases}$$

$$9.\ \text{REZULT} = \begin{cases} \dfrac{x^2(x^2+1)^3}{\sin x\cos x - 3н} + n!,\ при\ n \ge 1,\ x < 0\ та\ y < 0 \\ \dfrac{1}{\sin\dfrac{x}{2}\sqrt{\cos^3\dfrac{x}{2}+y}} + \Phi(n),\ при\ n \ge 1,\ x > 0\ та\ y > 2 \\ R - в\ останніх\ випадках \end{cases}$$

$$10.\ \text{REZULT} = \begin{cases} \dfrac{\sin^2 x}{1-tgx} + \dfrac{1}{\cos^2 x+\sin^2 x} + \Phi(n),\ при\ n \ge 1,\ x < 0\ та\ y < 0 \\ \dfrac{x^3-6y^2+11-R}{(x-2)^2} + \dfrac{\sin^3 x}{\cos^4 x} + n!,\ при\ n \ge 1,\ x < 0\ та\ y < 0 \\ R - в\ останніх\ випадках \end{cases}$$

Завдання для самостійної роботи 3

№	Обчислювані значення	Вхідні параметри
1	$y = a*sin^2(b)+b*cos(a)^2;\ a = \sqrt[3]{\lvert b + c\rvert};$ $b = \sqrt{x}$	x, c
2	$y=a^2 + b^2\ ;\ a=ln\lvert x\rvert;$ $b=e^k+a$	x, k
3	$y=e^x+4^c;\ \ c=a^2+\sqrt{b};$ $a = b^3+ ln\lvert b\rvert$	x, b
4	$y=sin^3(a)+cos^2(x);\ \ a=c+k^2;$ $a=sin^5 b$	b, c
5	$y = a*cos(x)-b*sin(x);\ x= \sqrt[3]{a - b};$ $a=n^2 b$	n, b
6	$y = z^3+cos^2(r);\ z = cos^2(a);$ $r = \sqrt{a + \lvert x\rvert}$	x, k
7	$y = sin^4(a^2+ b^2);\ a=\sqrt{b + t};$ $t = b^2 + k^3$	b, k
8	$y =lg^2\lvert x + a\rvert;\ \ x = \sqrt{a + b};$ $a = e^{t+b}$	t, b
9	$y=sin^3(a+b);\ a=n^3+\sqrt{b};$ $b=lg^2\lvert x\rvert$	x, n
10	$y = arctg^3(x^2);\ x=p+k;$ $k= \sqrt{p + t^2}$	t, p
11	$y = {c^3}/{\cos(c)}\ ;\ c = a^2+b^2;$ $a = \sqrt{\lvert x\rvert}+e^{\sqrt{b}}$	x, b
12	$y = ln\lvert x + t\rvert;\ x = t^2+p;$ $t= \sqrt{m}$	m, p
13	$y = \sqrt{x}\, sin(a) + a\sqrt{b} cos x;\ a= lg\lvert x\rvert;$ $b = x+p^3$	x, pb
14	$y = arctg^2\lvert x\rvert;\ x = t^3+b^2;\ \ t = b^3+ e^{\sqrt{q}}$	q, b
15	$y = cos3(x)+ \lvert a\rvert;\ x = e^b$ $b = \lvert a\rvert+\sqrt{a + p}$	a, p

Контрольні питання

1. Що таке функція?
2. Для чого використовуються функції?
3. Що таке прототип функції?
4. Яким чином виконується виклик функції?
5. Значення яких типів може повертати функція?
6. Яким чином повернути значення з функції?
7. Які типи можуть мати параметри функції?
8. Яким чином можна передати у функцію в якості аргументу масив?
9 Для чого використовуються посилання?
10. Які існують варіанти передачі аргументів у функцію?
11. Що таке рекурсивна функція?

СПИСОК ЛІТЕРАТУРИ

1. Васильченков О.Г., Тверитникова О.Є., Крылова В.А. Основы алгоритмизации и программирования на С++. Учебное пособие. Харьков: НТУ "ХПІ", 2015. 228 с.

2. Тверитникова О.Є., Крылова В.А. Методические указания по информатике «Основы программирования на С++». Харьков: НТУ “ХПІ”, 2013. 52 с.

3. Тверитникова О Є., Крылова В. А., Опрышкина М.И Методические указания по информатике «Основы программирования на С++». Харьков: НТУ “ХПІ”, 2014. 68 с.

4. Тверитникова О.Є., Крылова В.А. Методические указания по информатике «Основы программирования на С++». Двухмерные массивы. Харьков: ТУ «ХПІ», 2017. 52 с.

5. Ткачук В.М. Програмування на С++. Лабораторний практикум Івано-Франківськ: Видавництво Прикарпатського національного університету імені Василя Стефаника, 2011. 160 с.

6. Віник В.Ю. Алгоритмічні мови та основи програмування: мова С. навчальний посібник. Житомір: ЖДТУ, 2007. 328 с.

7. Методичні вказівки до лабораторних робіт з курсів «Інформатика» / І.П. Голубєва та ін. Київ: НТУУ «КПІ», 2013. 124 с.

8. Льовкін В.М. Методичні вказівки до виконання лабораторних робіт з дисципліни “Основи програмування” для студентів спеціальності 121 “Інженерія програмного забезпечення” (всіх форм навчання). Запоріжжя: ЗНТУ, 2016. 64 с.

9. Павловская Т.А., Щупак Ю.А. Структурное программирование С/С++. Практикум. Санкт-Петербург: Питер, 2007. 239 с.

10. Основи програмування. Частина 1. Базові алгоритми. Методичні вказівки до лабораторних і практичних робіт / О.Г. Трофименко та ін. Одеса: ВЦ ОНАЗ 2014. 107 с.

11. С++. Теорія та практика: навч. посіб. з грифом МОНУ / О.Г. Трофименко та ін.; за ред. О. Г. Трофименко. Одеса: ВЦ ОНАЗ, 2011. 587 с.

12. С++. Основи програмування. Теорія та практика: підручник / О.Г. Трофименко та ін.; за ред. О.Г. Трофименко. Одеса: Фенікс, 2010. 544 с.

Навчальне видання

ТВЕРИТНИКОВА Олена Євгенівна
КРИЛОВА Вікторія Анатоліївна

ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ С++. ВИКОРИСТАННЯ ФУНКЦІЙ

Методичні вказівки

до виконання практичних та лабораторних робіт
для студентів спеціальностей
152 «Метрологія та вимірювальна техніка»
151 «Автоматизація та комп'ютерно-інтегровані технології»
172 «Телекомунікація та радіотехніка»

Відповідальний за випуск Качанов П.О.

Роботу до друку рекомендував О.В. Дудник

В авторській редакції

План 2020 р., поз.61

Підп. до друку _____. Формат 60×84 1/16. Папір офсетний.
Riso-друк. Гарнітура Times New Roman. Ум. друк. арк. 1,5
Наклад 50 прим. Зам. № _____. Ціна договірна.

Видавець Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Виготовлювач_____
_____