МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

М.О. Соболь, Н.Ю. Любченко, Ю.В. Паржин, Р.В. Пугачов

ОСНОВИ ПРОГРАМУВАННЯ НА C/C++ В ПРИКЛАДАХ

Частина 1

**Навчально-методичний посібник
для студентів комп’ютерних спеціальностей
вищих навчальних закладів**

Затверджено Вченою радою
НТУ “ХПІ”, протокол № 2
від 26.02.2021 року

Харків
НТУ “ХПІ”
2021

УДК 004.43(075)
О75

Рецензенти:

А.А. Коваленко, д-р техн. наук, професор, Харківський національний університет радіоелектроніки
Г.А. Кучук, д-р техн. наук, професор, Національний технічний університет "Харківський політехнічний інститут"

Автори: М.О. Соболь, к.т.н.
Н.Ю. Любченко, к.т.н., доцент
Ю.В. Паржин, д.т.н., доцент
Р.В. Пугачов, к.т.н., доцент

Затверджено Вченою радою НТУ “ХПІ”, протокол № 2 від 26.02.2021 року

Соболь М.О.
О75 Основи програмування на C/C++ в прикладах. Частина 1: навч.-метод. посібник / Соболь М.О., Любченко Н.Ю, Паржин Ю.В., Пугачов Р.В. – Харків : НТУ "ХПІ", 2021. – 113 с.

Навчально-методичний посібник складається з восьми практичних робіт в прикладах, які охоплюють першу частину курсу «Основи програмування», присвячені розділам початкового курсу програмування: від завдань консольного введення та виведення, використання циклів і операцій з масивами до роботи з функціями. Приклади представлені у вигляді добре документованих початкового програмного коду.

Для студентів комп’ютерних спеціальностей вищих навчальних закладів.

Іл. 2. Табл. 10. Бібліогр. 5 назв.

УДК 004.43(075)

ВСТУП

Для того, щоб стати програмістом, недостатньо прослухати курс лекцій або прочитати самовчитель по мові програмування, потрібно писати програми, вирішувати конкретні завдання. Однак, як стверджують професійні програмісти, писати код становить близько 30% від того, що потрібно зробити, щоб вивчити мову. Другі 30% складають експерименти з кодом, який пишеться, змінюючи цей код і реально розуміючи, як він працює. Останні 40% того, що робиться для вивчення мови, – це читання початкових кодів інших людей. Але де їх взяти? У підручниках, як правило, наводяться типові завдання. Вони, без сумніву, корисні, але їх замало.

У цьому навчально-методичному посібнику починаючому програмісту пропонуються завдання, які, з одного боку, йому по плечу, з іншого – корисні тому, що містять приклади їх вирішення.

Навчально-методичний посібник має практичну спрямованість, містить теоретичний матеріал, практичні завдання та методику їх виконання у вигляді початкового коду, складається з восьми практичних робіт, які охоплюють першу частину курсу «Основи програмування».

Для поліпшення результатів навчання необхідно дотримуватися наступних рекомендацій: по-перше, необхідно згадати теоретичній матеріал, який наведено на початку кожного практичного заняття; по-друге, після ознайомлення з завданням, яке потрібно вирішити, необхідно спробувати написати програму самостійно, і тільки після цього проаналізувати запропоноване рішення. Для аналізу запропонованого методу рішення необхідно активно використовувати інструменти середовища розробки для виконання програми по кроках і перегляду поточних значень змінних.

Наведені приклади зроблені в середовище розробки Visual Studio, але остаточний вибір середовища та компілятору залишається за читачем.

Практична робота 1

ОСНОВНІ АРИФМЕТИЧНІ ОПЕРАЦІЇ

Мета: Оволодіння практичними навичками роботи з основними арифметичними операціями мов С/С++

Теми для попереднього опрацьовування

- Арифметичні операції
- Порядок виконання операцій

Теоретичні відомості

У мові С представлені всі стандартні *арифметичні операції*: додавання (+), віднімання (–), множення (*), ділення (/) та ділення за модулем (%). Перші чотири операції не вимагають роз'яснень, хоча при діленні цілих чисел дробова частина не враховується. Вираз *X* % *Y* дає залишок від виконання операції цілочисельного ділення *X* на *Y* і, отже, дорівнює нулю, коли *X* ділиться на *Y* без залишку.

Приклад 1.

```
int ia = 3, ib = 8, id;
id = ib % ia;              // результат: 2
```

У мові С *операція присвоювання* (=) може входити до складу інших виразів. У результаті виконання операції присвоювання повертається значення, присвоєне лівому операнду в операторі присвоювання. Наприклад, наступний вираз цілком коректний:

```
iv = 8 * (iw = 5);          // iv = 40
```

У даному випадку спочатку змінній *iw* присвоюється значення 5, після чого це значення множиться на 8, а результат присвоюється змінній *iv*.

У мові С в операторі присвоювання можна використовувати як арифметичні, так і логічні операції. Існує також специфічна форма запису оператора присвоювання. В табл. 3.1 наведені приклади операторів присвоювання.

Таблиця 1.1 – Приклади операторів присвоювання

Стандартна форма оператора	Еквівалент	Коментар
`var = var + 3;`	`var += 3;`	До змінної *var* додається 3
`var = var – 10;`	`var –=  10;`	Із змінної *var* віднімається 10
`var = var * 3.14;`	`var *= 3.14;`	Змінна *var* множиться на 3.14
`var = var / 2.5;`	`var /= 2.5;`	Змінна *var* ділиться на 2.5
`var = var & 0xF;`	`var &= 0xF;`	Над змінною *var* і константою 0*xF* виконується логічна операція *&*
`var = var | 0xF;`	`var |= 0xF;`	Над змінною *var* і константою 0*xF* виконується логічна операція |
`var = var << 3;`	`var <<= 3;`	Значення змінної *var* зсувається вліво на 3 розряди
`var = var >> 5;`	`var >>= 5;`	Значення змінної *var* зсувається вправо на 5 розрядів
`var = var % 2;`	`var %= 2;`	Змінна *var* отримає значення залишку після ділення *var* на 2
`var = var + 1;`	`var ++;`	Операція інкремента
`var = var – 1;`	`var --;`	Операція декремента

Операції інкремента і декремента можуть бути постфіксними або префіксними. При постфіксній операції, коли позначення `++` або `--` стоїть після позначення змінної, що входить в якийсь вираз, у виразі використову-

ється поточне значення змінної і лише після цього виконується його модифікація. При префіксній операції, коли позначення ++ або -- стоїть перед позначенням змінної, спочатку виконується модифікація змінної, і лише потім у виразі використовується її нове значення. Наприклад:

```
int k = 3, i = 3;
k = ++i;          // i = 4, k = 4
k = i++;          // i = 5, k = 4
k = i--;          // i = 4, k = 5
k = --i;          // i = 3, k = 3
```

У мові С *вирази* складаються з операндів, знаків операцій і дужок та використовуються для обчислення деякого значення певного типу.

Кожен *операнд* є, у свою чергу, виразом або одним з його окремих варіантів – константою чи змінною.

Арифметичні вирази на мові С мають бути записані в один рядок. Для гарантованого виконання операцій у необхідній послідовності, вони виконуються відповідно до *пріоритетів* – правил старшинства операцій, які загалом збігаються з тими, що використовуються в алгебрі:

– операції у частинах виразів, які розміщені всередині круглих дужок, виконуються передусім;

– наступними виконуються операції множення, ділення і обчислення залишку. Якщо вираз містить послідовність декількох операцій множення, ділення або обчислення залишку, то вони виконуються у послідовності зліва направо. Наприклад, $a \cdot b \cdot c$ означає $(a \cdot b) \cdot c$;

– операції додавання і віднімання виконуються в останню чергу. Якщо вираз містить декілька операцій додавання і віднімання, то вони виконуються у послідовності зліва направо. Операції додавання і віднімання мають однаковий пріоритет.

Для того, щоб зробити вираз більш наочним, в нього можна включити додаткові зайві *круглі дужки***.**

Результат обчислення виразу характеризується типом і значенням. Наприклад, якщо *a* і *b* – змінні цілого типу і описані так:

```
int a = 2, b = 5;
```

то вираз $a + b$ має значення 7 і тип – *int*. В операторі $a = b$ вираз має

значення, роміщене в змінній *a* (в даному випадку 5) і тип, співпадаючий з типом цієї змінної.

Якщо змінній привласнюється значення, яке не входить в діапазон значень для даного типу змінної, виникає помилка переповнювання. При виконанні арифметичних операцій необхідно пам'ятати про можливість виникнення таких помилок.

Задачі

Загальні умови:

Кожне завдання містить приклад використання основних арифметичних операцій мов C/C++, і може містити умови вирішення. Необхідно з'ясувати, що буде отримано програмою, яка реалізує запропоновані операції і умови. Після списку завдань надано рішення у вигляді програмного коду.

1) `x = (7 + 6) % 5 / 2;`
2) `x = -3 * 4 % -6 / 5;`
3)
```
int x = 2, y = 1, z = 3;
x *=8 / 2 + y++ + ++z;
```
4)
```
m = 2; x = 2;
x *= y = z = 4/m;
```
5)
```
int x = 2, y, z;
    а)  x *= 3 + 2;
    б)  x += y = z = 4;
    в)  x = y == z;
    г)  x = x == (y = z);
```
6)
```
z = 1; x = 2; y = 3;
z = 3 * x++ - --y;
```
7) Дано двозначне число. Необхідно виділити в ньому одиниці і десятки.

8) Дано тризначне число. Визначити кількість сотень, десятків і одиниць в ньому. Написати число, отримане при прочитанні початкового числа з права на ліво.

9) Дано двозначне число. Знайти число, отримане переставленням першої і останньої цифр.

10) Дано тризначне число. У ньому закреслили першу зліва цифру і приписали її в кінець вихідного числа. Знайти число, що вийшло.

11) Дано тризначне число. У ньому закреслили останню цифру і приписали її в початок вихідного числа. Знайти число, що вийшло.

12) Дано тризначне число. З ним необхідно виконати наступні дії:

а) відняти останню цифру;

б) результат розділити на 10;

в) до результату зліва приписати останню цифру вихідного числа.

13) Дано тризначне число. Знайти число, отримане шляхом перестановки першої та другої цифри вихідного числа.

Вирішення задач 1 – 13:

```
#include <iostream>

int main()
{
	setlocale(LC_ALL, "Ukrainian");
	{// 1)
		int x = (7 + 6) % 5 / 2;	// x = 1
		//	1  2  3
		printf("1) x = %d \n", x);
	}
	{// 2)
		int x = -3 * 4 % -6 / 5;	// x = 0
		//	1  3  4  2 5
		printf("2) x = %d \n", x);
	}
	{// 3)
		int x = 2, y = 1, z = 3;
		x *= 8 / 2 + y++ + ++z;		// x = 18
		printf("3) x = %d \n", x);
		// буде: y = 2, z = 4, x = 18
	}
	{// 4)
		int m = 2, x = 2, y, z;
		x *= y = z = 4 / m;		// x = 4
		printf("4) x = %d \n", x);
		// буде: m = 2, x = 4, y = 2, z = 2
	}
	{// 5)
		int x = 2, y, z;
		x *= 3 + 2;			// а) x = 10
		x *= y = z = 4;			// б) x = 40
						// y = 4
```

```
                                    // z = 4
        x = y == z;                 // в) x = 1
        // 2   1
        x = x == (y = z);           // г) x = 0
        // (1 == 4)?                // false (=0)
        printf("5) г) x = %d \n", x);
}
{// 6)
        int z = 1, x = 2, y = 3;
        z = 3 * x++ - --y;          // z = 4
        printf("6) z = %d \n", z);
        // буде: x = 2, y = 2, z = 4
}
{// 7)
        short a = 52, dec = 0, ed = 0;
        dec = a / 10;               // dec = 5
        ed = a % 10;                // ed = 2
        a = dec * 10 + ed;          // перевірка
        printf("7) dec = %d \n", dec);
        printf("    ed = %d \n", ed);
}
{// 8)
        short a = 352, sot = 0, dec = 0, ed = 0;
        sot = a / 100;              // sot = 3
        dec = a % 100 / 10;         // dec = 5
        ed = a % 10;                // ed = 2
        a = ed * 100 + dec * 10 + sot;
        printf("8) a = %d \n", a);
}
{// 9)
        short a = 25, dec = 0, ed = 0;
        dec = a / 10;               // dec = 2
        ed = a % 10;                // ed = 5
        a = ed * 10 + dec;          // 52
        printf("9) a = %d \n", a);
}
{// 10)
        short a = 352, dec = 0, ed = 0;
        dec = a % 100;              // dec = 52
        ed = a / 100;               // ed = 3
        a = dec * 10 + ed;          // a = 523
        printf("10) a = %d \n", a);
}
{// 11)
        short a = 352, dec = 0, ed = 0;
```

```
        dec = a / 10;                // dec = 35
        ed = a % 10;                 // ed = 2
        a = ed * 100 + dec;          // a = 235
        printf("11) a = %d \n", a);
    }
    {// 12)
        short x = 352, dec = 0, ed = 0;
        // а)
        ed = x % 10;                 // ed = 2
        x = x - ed;                  // x = 350
        // б)
        x = x / 10;                  // x = 35
        // в)
        x = ed * 100 + x;            // x= 235
        // або
        x = 352;
        x = (x % 10) * 100 + (x - x % 10) / 10;
        printf("12) x = %d \n", x);
    }
    {// 13)
        short x = 352, sot = 0, dec = 0, ed = 0;
        sot = x / 100;               // sot = 3
        dec = x % 100 / 10;          // dec = 5
        ed = x % 10;                 // ed = 2
        x = dec * 100 + sot * 10 + ed;      // x = 532
        // або
        x = 352;
    x = (x % 100 / 10) * 100 + (x / 100) * 10 + x % 10;
        printf("13) x = %d \n", x);
    }
    return 0;
}
```

Практична робота 2

ЛОГІЧНІ ОПЕРАЦІЇ ТА ПОРІВНЯННЯ

Мета: Оволодіння практичними навичками роботи з логічними операціями мов С/С++, та операціями порівняння

Теми для попереднього опрацьовування

- Логічні операції
- Оператори порівняння

Теоретичні відомості

Логічні операції

У мові С існують наступні логічні операції: логічне множення "І" – кон`юнкція (`&&`), логічне додавання "АБО" – диз`юнкція (`||`), логічне заперечення "НІ" (`!`). Значенням логічного виразу є хибність (*false* або нуль) чи істина (*true* або одиниця).

Таблиці істинності логічних операцій наведені в табл. 2.1.

Таблиця 2.1 – Таблиці істинності логічних операцій

Операнд 1	Операнд 2	Кон`юнкція `&&`	Диз`юнкція `||`	Заперечення `!` (операнда 1)
хибність	хибність	хибність	хибність	істина
хибність	істина	хибність	істина	істина
істина	хибність	хибність	істина	хибність
істина	істина	істина	істина	хибність

Приклад 1.

Логічне І: `(x >= 1) && (x <= 500)`

Логічне АБО: `(input == 'q') || (input == 'Q')`

Логічне заперечення: `!(x > y && y > z)`

Для ефективності в мові С логічний вираз обчислюється зліва направо і припиняється подальше його обчислення, коли значення уже очевидне.

Приклад 2.

```
(x < y) && (y < z)
```

Цей вираз істинний, якщо істинні як вираз ($x < y$), так і вираз ($y < z$). Якщо ($x < y$) хибність, то весь вираз має бути помилковим і використання значення виразу ($y < z$) ігнорується.

Приклад 3.

```
(x < y) || (y < z)
```

Цей вираз істинний, якщо істинний або вираз ($x < y$), або вираз ($y < z$), або якщо істинні обидва вирази. Якщо ($x < y$) істинний, весь вираз істинний, так що в цьому випадку не обчислюється істинність виразу ($y < z$).

Унарна операція ! ("НІ") змінює на протилежне початкове логічне значення. Якщо ($x < y$) хибність, то !($x < y$) істина.

Оператори порівняння

У мові С є шість операторів порівняння, які наведені в табл. 2.2.

Таблиця 2.2 – Оператори порівняння

Оператор	Символ	Приклад	Операція
Більше	`>`	`x > y`	*true*, якщо *x* більше *y*, в іншому випадку – *false*
Менше	`<`	`x < y`	*true*, якщо *x* менше *y*, в іншому випадку – *false*
Більше або дорівнює	`>=`	`x >= y`	*true*, якщо *x* більше / дорівнює *y*, в іншому випадку – *false*
Менше або дорівнює	`<=`	`x <= y`	*true*, якщо *x* менше / дорівнює *y*, в іншому випадку – *false*
Дорівнює	`==`	`x == y`	*true*, якщо *x* дорівнює *y*, в іншому випадку – *false*
Не дорівнює	`!=`	`x != y`	*true*, якщо *x* не дорівнює *y*, в іншому випадку – *false*

Ви вже могли їх бачити в коді. Вони досить прості. Кожен з цих операторів обчислюється в логічне значення *true* (1) або *false* (0).

Задачі

Загальні умови:

Кожне завдання містить приклад використання логічний операції та операторів порівняння мов C/C++, і може містити умови вирішення. Необхідно з'ясувати, що буде отримано програмою, яка реалізує запропоновані операції і умови. Після списку завдань надано рішення у вигляді програмного коду.

1) `x = 2, y = 1, z = 0;`
   а) `x = x && y || z;`
   б) `x || ! y && z;`
   в) `x = y = 1; z = x++ - 1;`
   г) `z += -x++ + ++y;`
2) Визначити значення логічного виразу, якщо `A = 1, B = 0, C = 0`:
   а) `A || B && !C;`
   б) `!A && !B;`
   в) `!(A && B) || B;`
   г) `A && !B || C;`
   д) `A && (!B || C);`
   е) `A && (!(B || C)).`
3) Визначити значення логічного виразу, якщо `x = 0, y = 1, z = 0`:
   ж) *x* та не (*z* або *y*) або не *z*;
   з) (не *x* або не *y*) та (*x* або *y*);
   и) *x* та *y* або *x* та *z* або не *z*.
4) Визначити значення логічного величини при всіх можливих значеннях логічних величин x і y (побудувати таблицю істинності):
   к) `!x && !y;`
   л) `y || !(x && !y);`
   м) `F = !(x || !y) || !y.`
5) Записати логічний вираз, який має значення *true*, тільки при виконанні зазначених умов:
   н) `x > 2` та `y > 2;`
   о) `x > 1` або `y > -2;`
   п) невірно, що `x > 2;`
   р) невірно, що `x > 0` та `x < 5;`
   с) `10 < x <= 20;`

т) `x < 5 або x > 10;`

у) `0 < y <= y та x > 5.`

6) Записати логічний вираз, який має значення *true*, якщо:

а) кожне з чисел *x* та *y* більше 100;

б) тільки одне з *x* та *y* парне;

в) хоча б одне з *x* та *y* більше 0;

г) кожне з чисел *x*, *y*, *z* кратне 3;

д) тільки одне з *x*, *y*, *z* менше 0;

е) *x* кратне 2 або 3;

ж) *x* не кратне 3 та закінчується на 0.

Вирішення задач 1 – 6:

```
#include <iostream>

int main()
{
	setlocale(LC_ALL, "Ukrainian");
	// 1)
		int x = 2, y = 1, z = 0;
		// а)
		x = x && y || z;			// x = 1
		// б)
		x = x || !y && z;			// x = 1
		// в)
		x = y = 1;
		z = x++ - 1;				// x = 2
							// y = 1
							// z = 0
		// г)
		z += -x++ + ++y;			// x = 3
							// y = 2
							// z = 0;
	// 2)
		bool A = 1, B = 0, C = 0, Result = 0;
		// а)
		Result = A || B && !C;		// Result = 1
		// б)
		Result = !A && !B;			// Result = 0
		// в)
		Result = !(A && B) || B;		// Result = 1
		// г)
```

```
            Result = A && !B || C;       // Result = 1
            // д)
            Result = A && (!B || C);     // Result = 1
            // е)
            Result = A && (!(B || C));   // Result = 1
      // 3)
            bool X = 0, Y = 1, Z = 0;
            // а)
            Result = X && !(Z || Y) || !Z;      // Result = 1
            // б)
            Result = (!X || !Y) && (X || Y);
            // в)
            Result = X && Y || X && Z || !Z;
      // 4)
            printf("4) \n");
            // а)  !x && !y
            printf("а) !X && !Y \n");
            printf("| X | Y | !X | !Y | !X && !Y |\n");
            X = 0; Y = 0;
            printf("| %d | %d |  %d |  %d |     %d    |\n",
                  X, Y, !X, !Y, !X && !Y);
            X = 0; Y = 1;
            printf("| %d | %d |  %d |  %d |     %d    |\n",
                  X, Y, !X, !Y, !X && !Y);
            X = 1; Y = 0;
            printf("| %d | %d |  %d |  %d |     %d    |\n",
                  X, Y, !X, !Y, !X && !Y);
            X = 1; Y = 1;
            printf("| %d | %d |  %d |  %d |     %d    |\n",
                  X, Y, !X, !Y, !X && !Y);
            // б)  y || !(x && !y)
            printf("б) Y || !(X && !Y) \n");
      printf("|X |Y | !Y |X && !Y | !(X&&!Y) |Y ||!(X&&!Y) |");
            X = 0; Y = 0;
      printf("| %d | %d |  %d |    %d    |     %d    |       %d
|\n", X, Y, !Y, X && !Y, !(X && !Y), Y || !(X && !Y));
            X = 0; Y = 1;
      printf("| %d | %d |  %d |    %d    |     %d    |       %d
|\n",  X, Y, !Y, X && !Y, !(X && !Y), Y || !(X && !Y));
            X = 1; Y = 0;
      printf("| %d | %d |  %d |    %d    |     %d    |       %d
|\n",  X, Y, !Y, X && !Y, !(X && !Y), Y || !(X && !Y));
            X = 1; Y = 1;
      printf("| %d | %d |  %d |    %d    |     %d    |       %d
|\n",  X, Y, !Y, X && !Y, !(X && !Y), Y || !(X && !Y));
```

```
            // в)  !(x || !y) || !y
            printf("в) F = !(X || !Y) || !Y \n");
      printf("| X | Y | !Y | X || !Y | !(X||!Y) | F |\n");
            X = 0; Y = 0;
            printf("| %d | %d |  %d |    %d    |     %d    |
%d |\n",X, Y, !Y, X || !Y, !(X || !Y), !(X || !Y) || !Y);
            X = 0; Y = 1;
            printf("| %d | %d |  %d |    %d    |     %d    |
%d |\n",X, Y, !Y, X || !Y, !(X || !Y), !(X || !Y) || !Y);
            X = 1; Y = 0;
            printf("| %d | %d |  %d |    %d    |     %d    |
%d |\n",X, Y, !Y, X || !Y, !(X || !Y), !(X || !Y) || !Y);
            X = 1; Y = 1;
            printf("| %d | %d |  %d |    %d    |     %d    |
%d |\n",X, Y, !Y, X || !Y, !(X || !Y), !(X || !Y) || !Y);
      // 5)
            // а)  x > 2 та y > 2;
            x = 3; y = 3;
            Result = (x > 2) && (y > 2);
            // б)  x > 1 або y > -2;
            x = 3; y = -3;
            Result = (x > 1) || (y > -2);
            // в)  невірно, що x > 2;
            x = 0;
            Result = !(x > 2);
            // г)  невірно, що x > 0 та x < 5;
            x = 0;
            Result = !(x > 0 && x < 5);
            // д)  10 < x <= 20;
            x = 20;
            Result = (x > 10 && x <= 20);
            // е)  x < 5 або x > 10;
            x = 0;
            Result = (x < 5 || x > 10);
            // ж)  0 < y <= y та x > 5;
            x = 10; y = 4;
            Result = (y > 0 && y <= 4) && (x > 5);
      // 6)
            // а)  кожне з чисел x та y більше 100;
            x = 105, y = 105;
            Result = (x > 100) && (y > 100);
            // б)  тільки одне з x та y парне;
            x = 106, y = 105;
            Result = (x % 2 == 0) || (y % 2 == 0);
            // в)  хоча б одне з x та y більше 0;
```

```
        x = 106, y = -105;
        Result = x > 0 || y > 0;
        // г)  кожне з чисел x, y, z кратне 3;
        x = 6, y = 9, z = 12;
    Result = (x % 3 == 0) && (y % 3 == 0) && (z % 3 == 0);
        // д)  тільки одне з x, y, z менше 0;
        x = 6, y = 9, z = -12;
        Result = (x < 0) || (y < 0) || (z < 0);
        // е)  x кратне 2 або 3;
        x = 9;
        Result = (x % 2 == 0) || (x % 3 == 0);
        // ж)  x не кратне 3 та закінчується на 0;
        x = 10;
        Result = (x % 3 != 0) && (x % 10 == 0);

        return 0;
}
```

Практична робота 3

ПОБІТОВІ ОПЕРАЦІЇ

Мета: Оволодіння практичними навичками роботи побітовими операціями і операціями зсуву мов С/С++

Теми для попереднього опрацьовування

- Побітові операції
- Операції зсуву

Теоретичні відомості

Побітові операції

Побітові операції (див. таблиці) призначені для дії на окремі біти даних. Ці операції використовуються в тих програмах, які забезпечують управління апаратурою на фізичному рівні, оскільки в регістрах апаратури кожен біт має певну функціональну орієнтацію. Інша область використання побітових операцій – робота з ознаками режиму або того чи іншого призначення. Числа – сукупності значень ознак – широко використовуються в операційних системах. Побітові операції застосовні тільки до *цілочисельних* значень. Перелік побітових операцій наведений в табл. 3.1.

Таблиця 3.1 – Побітові операції

Операція	Призначення	Операція	Призначення
~	НІ (заперечення)	|	АБО
&	І	>>	Зсув вправо
^	Виключне АБО	<<	Зсув вліво

Операція *побітового заперечення НІ* (~) інвертує двійковий код числа, тобто всі двійкові одиниці замінює на двійкові нулі і навпаки (див. табл. 3.2).

Таблиця 3.2 – Операція побітового заперечення

Початкове число$_{16}$	9	2	C	4
Початкове число$_{2}$	1001	0010	1100	0100
Результат$_{2}$	0110	1101	0011	1011
Результат$_{16}$	9	D	3	B

Операція *побітового І* (&) виконується над двома операндами. У остаточному двійковому коді встановлюються в одиницю тільки ті двійкові розряди, в яких в обох операндах встановлені одиниці. Якщо хоч би в одному з операндів в деякому розряді є нуль, то і в результаті в цьому розряді буде також нуль.

Зазвичай один з операндів є початковим числом, а другий – маскою, за допомогою якої виділяються необхідні біти операнда. В табл. 3.3 представлене виділення 4 старших бітів шістнадцяткового числа $92C4_{16}$.

Таблиця 3.3 – Операція побітового І (виділення 4 старших бітів)

Початкове число$_{16}$	9	2	C	4
Початкове число$_2$	1001	0010	1100	0100
Маска$_2$	1111	0000	0000	0000
Результат$_2$	1001	0000	0000	0000
Результат$_{16}$	9	0	0	0

Операція *побітового АБО* (|) записує в біт результату одиницю в тому випадку, коли хоча б один з бітів, номери яких відповідають номеру біта результату, дорівнює одиниці.

Операція *побітового АБО* також виконується над двома операндами, і також зазвичай один з них є початковим числом, а другий – маскою, за допомогою якої в початковому числі з'являться або підтвердяться одиниці в тих двійкових розрядах, які встановлені в масці. В табл. 3.4 двійкові одиниці за допомогою маски додаються в 0, 1 і 15 біт шістнадцяткового числа $92C4_{16}$. Оскільки розряд 15 в числі і так був встановлений, його стан не змінюється.

Таблиця 3.4 – Операція побітового АБО (встановлення 0, 1 та 15 бітів)

Початкове число$_{16}$	9	2	C	4
Початкове число$_2$	1001	0010	1100	0100
Маска$_2$	1000	0000	0000	0011
Результат$_2$	1001	0010	1100	0111
Результат$_{16}$	9	2	C	7

Операція *побітового виключного АБО* (^) записує в біт результату одиницю в тому випадку, якщо порівнювані біти відрізняються один від одного.

Операція *побітового виключного АБО* відрізняється від звичайного *АБО* тим, що за наявності в деякому двійковому розряді одиниці в обох операндах відповідний розряд результату скидається в нуль. В табл. 3.5 наведено приклад використання *побітового виключного АБО*.

Таблиця 3.5 – Операція побітового виключного АБО

Початкове число$_{16}$	9	2	C	4
Початкове число$_{2}$	1001	0010	1100	0100
Маска$_{2}$	1000	0000	0000	0011
Результат$_{2}$	0001	0010	1100	0111
Результат$_{16}$	1	2	C	7

Операції зсуву

Операції *зсуву вліво* (<<) та *зсуву вправо* (>>) полягають в зсуві двійкового подання *цілочисельної* змінної, ім'я якої вказане зліва від операції зсуву, вліво(вправо) на кількість бітів, вказану праворуч від операції. При зсуві вліво звільнені молодші біти заповнюються нулями, а попередні значення відповідної кількості старших бітів втрачаються.

Зсув беззнакового числа на одну позицію вліво еквівалентне множенню числа на 2. Зсув беззнакового числа на одну позицію вправо еквівалентне діленню числа на 2.

В табл. 3.6 наведено приклад використання операції зсуву числа без знаку вліво на два розряди.

Таблиця 3.6 – Операція зсуву числа без знаку вліво на два розряди

Початкове число$_{16}$	9	2	C	4
Початкове число$_{2}$	1001	0010	1100	0100
Результат$_{2}$	0100	1011	0001	0000
Результат$_{16}$	4	B	1	0

Задачі

Загальні умови

Кожне завдання містить приклад використання побітових операції мов C/C++, і може містити умови вирішення. Необхідно з'ясувати, що буде отримано програмою, яка реалізує запропоновані операції. Після списку завдань надано рішення у вигляді програмного коду.

1) Визначити значення виразу, якщо `x = 3, y = 2, z = 1`:

```
а) x | y & z;
б) x | y & ~z;
в) x ^ y & ~z;
г) x & y && z.
```

2) Визначити значення виразу, якщо `x = 1`:

```
а) !x | x;
б) ~x | x;
в) x ^ x;
г) x << 3.
```

3) Визначити значення виразу, якщо `x = 5, y = 6, z = 7`:

```
x & y | z.
```

4) Нехай `x = 9`. Знайти чому буде дорівнювати вираз:

```
а) x << 3;
б) x >> 3;
в) x >> 5.
```

5) Нехай `x = 255`. Знайти чому буде дорівнювати x після *послідовного* виконання наступних операцій

```
а) x = x << 3;
б) x = x >> 3.
```

6) Для `x = 7`, для числа, представленого як беззнаковий однобайтовий тип виконати послідовність операцій

```
x = 7;
x = x << 1;
x = x << 3;
x = x << 2;
x = x >> 1;
x = x >> 2.
```

7) Для `x = 7`, для числа, представленого як знаковий однобайтовий тип виконати ту саму послідовність операцій, що і в п.6

Вирішення задач 1 – 7:

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

int main()
{
	setlocale(LC_ALL, "Ukrainian");
	{// 1
		int x = 3;
		int y = 2;
		int z = 1;
		// 1 а)
		printf("1 а) %d \n", x | y & z);		//3
		//                      2   1
		// 1 б)
		printf("1 б) %d \n", x | y & ~z);		//3
		//                      3   2 1
		// 1 в)
		printf("1 в) %d \n", x ^ y & ~z);		//1
		//                      3   2 1
		// 1 г)
		printf("1 г) %d \n", x & y && z);		//1
		//                      1    2
		// 1 г) другий варіант друку результату
		x = x & y && z;
		printf("1 г) %s \n", x ? "true" : "false");//true
	}
	{// 2
		int x = 1;
		// 2 а)
		printf("2 а) %d \n", !x | x);			//1
		//                   1  2
		// 2 б)
		printf("2 б) %d \n", ~x | x);			//-1
		//                   1  2
		// 2 в)
		printf("2 в) %d \n", x ^ x);		//0
		// 2 г)
		printf("2 г) %d \n", x << 3);			//8
	}
	{// 3)
		int x = 5, y = 6, z = 7;			//7
		printf("3) x = %d \n", x & y | z);
		//                      1   2
```

```
	}
	{// 4
		int x = 9;
		// 4 а)
		printf("4 а) %d \n", x << 3);		//72
		// 4 б)
		printf("4 б) %d \n", x >> 3);		//1
		// 4 в)
		printf("4 в) %d \n", x >> 5);		//0
	}
	{// 5
		int8_t x = 52;
		// 5 а)
		x = x << 3;
		printf("5 а) %d \n", x);		//-96
		// 5 б)
		x = x >> 3;
		printf("4 б) %d \n", x);		//-12
	}
	{// 6
		uint8_t  x;
		x = 7;
		x = x << 1; printf("x = %d \n", x); //14
		x = x << 3; printf("x = %d \n", x); //112
		x = x << 2; printf("x = %d \n", x); //192
		x = x >> 1; printf("x = %d \n", x); //96
		x = x >> 2; printf("x = %d \n", x); //24
	}

	{// 7
		int8_t x;
		x = 7;
		x = x << 1; printf("x = %d \n", x); //14
		x = x << 3; printf("x = %d \n", x); //112
		x = x << 2; printf("x = %d \n", x); //-64
		x = x >> 1; printf("x = %d \n", x); //-32
		x = x >> 2; printf("x = %d \n", x); //-8
	}

	return 0;
}
```

Практична робота 4

УМОВНІ ОПЕРАТОРИ

Мета: Отримання практичних навичок в роботі з умовним оператором і розгалуженими алгоритмами в мовах C/C++.

Теми для попереднього опрацьовування

- Логічні операції
- Оператори вибору

Теоретичні відомості

Оператори вибору

У мові С є група операторів, що дозволяють здійснювати переходи за певними умовами і вибирати варіанти.

Оператор ***if*** *і операції відношення*

Оператор *if* дозволяє проаналізувати вказану умову і здійснити виконання деякого фрагмента програми тільки при істинному результаті аналізу. Оператора *if* має дві форми:

```
if (вираз) оператор1;
```

та

```
if (вираз) оператор1; else оператор2;
```

Якщо в результаті обчислення виразу набуте значення *true* (на схемі алгоритму, наведеному на рис. 4.1, напрямок "*так*"), то в обох формах оператора *if* виконується `оператор1`. Якщо в результаті обчислення виразу набуте значення *false* (на схемі алгоритму (див. рис. 4.1) напрямок "*ні*"), тоді виконання оператора *if*, представленого в першій формі, закінчується, а в операторі, що має другу форму, виконується `оператор2`.

Якщо операторна частина гілки *if* або *else* містить не один вираз, а декілька, необхідно розмістити їх у фігурних дужках. Після закриваючої фігурної дужки крапка з комою не ставиться.

Оператори *if* першої і другої форми реалізують алгоритми представлені на рис. 4.1.

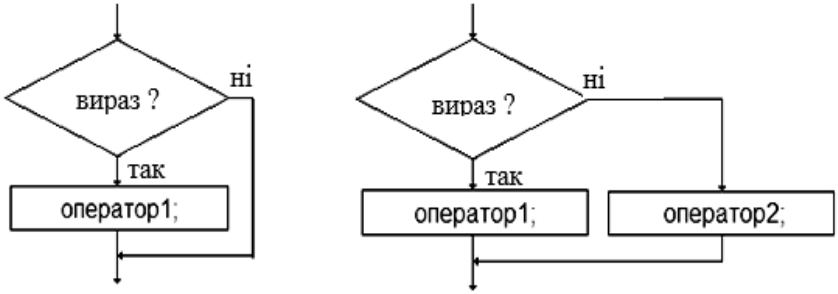

Рисунок 4.1 – Алгоритми роботи операторів ***if***

У *виразі* оператора *if* часто використовується та чи інша з операцій відношення, наведених в табл. 4.1.

Таблиця 4.1 – Операції відношення

Операція	Опис	Операція	Опис
= =	Дорівнює	> =	Більше або дорівнює
>	Більше	< =	Менше або дорівнює
<	Менше	! =	Не дорівнює

Одночасно з операціями відношення в операторі *if* можуть використовуватись логічні операції, що дає можливість створювати комбіновані конструкції перевірки даних.

Якщо у виразі оператора *if* присутній тільки арифметичний вираз, то значенням виразу оператора *if* буде *false*, якщо значення арифметичного виразу дорівнює нулю. При всіх інших значеннях арифметичного виразу це значення буде *true*.

Приклад 1. Різні форми оператора *if*

```
if  (x > largest)
   largest = x;
```

```
if (quantity < 1000)
  printf ("Кількість недостатня \n");
else
    printf ("Кількість достатня \n");

if ( x != 0)
     y = y/x;
else {
     printf ("Помилка");
     printf ("Ділення на нуль\n");
     y = 0;
}
```

Приклад 2. Одиночний оператор і блок (тобто складений оператор)

```
#include <stdio.h>
main( ){
      int i;
      i=getchar( );
      if(i > 5)  /* одиночний оператор*/
      printf("Значення більше 5.\n");
      if(i < 5) {    /* складений оператор*/
             printf("Значення \n");
             printf(" менше 5.\n");
      }
}
```

Оператор ***switch...case***

Оператор *switch...case* дозволяє залежно від значення деякого виразу вибрати один з багатьох варіантів продовження програми. Оператор має наступний формат:

```
switch(вираз){
      case(константний вираз1): оператор1; break;
      case(константний вираз2): оператор2; break;
...
      case(константний виразN): операторN; break;
      default: операторN+1;
}
```

У виразі оператора *switch* зазвичай використовується змінна типу *int* або *char*, хоча можна використовувати і складніші вирази, в які входять,

наприклад, арифметичні або логічні операції над декількома змінними і константами.

У *константному виразі* оператора *case* зазвичай використовуються просто константи (у числовій формі або в символьній, якщо вони були заздалегідь визначені за допомогою оператора препроцесора *#define*), проте можуть використовуватися і вирази над константами.

Виконання оператора *switch* починається з обчислення виразу, записаному в дужках в ньому, який повинен давати *цілочисельний* результат. Цей результат послідовно порівнюється з результатами обчислення константних виразів при операторах *case*, і, якщо буде виявлена рівність результатів, то виконується оператор відповідного *case*. Якщо збіг результатів не виявлений, виконується оператор при операторі *default*. Якщо оператор *default* відсутній, то починають виконуватися оператори, наступні за всією конструкцією *switch...case*.

Оператор *switch...case* реалізує алгоритм представлений на рис. 4.2.

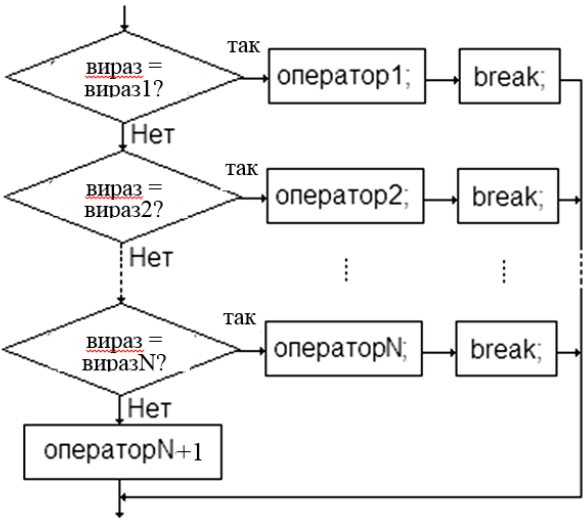

Рисунок 4.2 – Алгоритм роботи оператора *switch...case*

Приклад 3. Використання оператора *switch...case*

```
#include <stdio.h>
int main()
{
    int k;
    scanf("%d",&k);
    switch(k)    {
       case 0; printf("вибір 0\n");break;
       case 1; printf("вибір 1\n");break;
       case 2; printf("вибір 2\n");break;
       case 3; printf("вибір 3\n");break;
       default: printf("default\n");
    }
}
```

*Операція **умови:*** **? :**

У мові С є один короткий спосіб запису оператора *if-else*. В ньому використовуються знаки операцій – ? та :.

Розглянемо, як можна знайти абсолютне значення числа:

```
y < 0 ? x = -y : x = y;
```

Якщо *y* менше нуля, то `x = -y`, інакше `x = y`. В термінах оператора *if-else* даний оператор виглядав би так:

```
if (y < 0) x = -y;
else x = y;
```

У загальному вигляді такий оператор можна записати таким чином:

```
умовний вираз ? вираз1 : вираз2;
```

Якщо умовний вираз – істина, то значенням всього умовного оператора є значення виразу1**,** якщо – хибність, то значенням всього умовного оператора є значення виразу2.

Часто у виразі1 і виразі2 використовується одна і та ж змінна, якій привласнюється або одне, або інше значення. Тоді такий оператор записується трохи інакше:

```
змінна = умовний вираз ? вираз1 : вираз2;
```

Наприклад,

```
cS = x <= max ? 'Y' : 'N';
```

Якщо *x* <= *max*, то змінна *cS* набуває значення “Y”; якщо *x* > *max*, то *cS* набуває значення “N”.

Задачі

Загальні умови

Кожне завдання містить приклад використання умовних операцій мов C/C++, і може містити умови вирішення. Необхідно з'ясувати, що буде отримано програмою, яка реалізує запропоновані дії. Після списку завдань надано рішення у вигляді програмного коду.

1) Нехай дано значення змінної *a*. Написати умовний оператор, який обчислює наступні змінні *f*:

а) $f = \begin{cases} a^2, & \text{якщо } -2 \le a \le 2 \\ 4, & \text{в інших випадках} \end{cases}$

б) $f = \begin{cases} a^2 + 4a + 5, & \text{якщо } a \le 2 \\ 1/(a^2 + 4a + 5), & \text{в інших випадках} \end{cases}$

в) $f = \begin{cases} 0, & \text{якщо } a \le 0 \\ a, & \text{якщо } 0 \le a \le 1 \\ a^4, & \text{в інших випадках} \end{cases}$

Вирішення задачі 1:

```
// Обчислення змінної f
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
```

```
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int a;         // число, введене з клавіатури
        int f;         // результат обчислення змінної f

        printf("\nОбчислення значення змінної f\n ");
        printf("Введіть ціле число і натисніть <Enter>");
        printf("-> ");
    {//1а
        scanf("%i", &a);
        if (a >= -2 && a < 2)
               f = a *a;
        else
               f = 4;
        printf("Змінна f = %d ", f);
        }
    {//1б
        scanf("%i", &a);
        if (a <= 2)
               f = a *a + 4 * a +5;
        else
               f = 1 / (a *a + 4 * a +5);
        printf("Змінна f = %d ", f);
        }

    {//1в
        scanf("%i", &a);
        if (a <= 0)
               f = 0;
        else
               if (a <= 1)
                      f = a;
               else
                      f = pow(a, 4);
        printf("Змінна f = %d ", f);
        }
        return 0;
}
```

---

2) Нехай дані значення змінних *a, b, c*. Написати умовний оператор, який виконує наступні дії:

– якщо *a* <= *b* <= *c*, то всі числа записати їх квадратами;

– якщо *a* > *b* > *c*, то кожне замінити максимальним з трьох;

– в інших випадках – змінити знак кожного числа.

Вирішення задачі 2:

```
// Написати умовний оператор
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int a, b, c;  // числа, введені з клавіатури
	printf("Введіть цілі числа a, b, c і натисніть <Enter>");
	printf("-> ");
	scanf("%i %i %i", &a, &b, &c);
	if (a <= b && b <= c) {
		a = pow(a, 2);
		b = pow(b, 2);
		c = pow(c, 2);
	}
	else
		if (a > b && b > c) {
			b = a;
			c = a;
		}
		else {
			a = - a;
			b = - b;
			c = - c;
		}
	printf("a = %i  b = %i  c = %i", a, b, c);
	return 0;
}
```

3) Нехай дані значення змінних *x, y*. Написати умовний оператор, який виконує наступні дії:

– якщо $x < 0$ та $y < 0$, то кожне замінити його модулем;

– якщо негативне тільки одне, то обидва значення збільшити на 0,5;

– якщо обидва значення не негативні і жодне з них не належить відрізку [0,5 ; 2,0], то обидва значення зменшити в 10 разів;

– в інших випадках – *x, y* залишити без зміни.

Вирішення задачі 3:

```
// Написати умовний оператор
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	float x, y;   // числа, введені з клавіатури
	printf("Введіть цілі числа x, y і натисніть <Enter>");
	printf("-> ");	scanf("%f %f", &x, &y);
	if (x < 0 && y < 0) {
		x = fabs(x);
		y = fabs(y);
	}
	else
		if (x < 0 || y < 0) {
			x += 0.5;
			y += 0.5;
		}
		else
			if (!((x >= 0.5 && x <= 2.0) ||
			      (y >= 0.5 && y <= 2.0))){
			       x /= 10.0;
			       y /= 10.0;
			}
	printf("x = %lf  y = %lf", x, y);
	return 0;
}
```

4) Написати програму обчислення площі кільця. Програма повинна перевіряти правильність вихідних даних. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Обчислення площі кільця
Введіть данні:
радіус кільця (см) -> 3.5
радіус отвору (см) -> 7
```

```
    Помилка! Радіус отвору не може бути більше радіусу кі-
льця
```

Вирішення задачі 4:

```
// Обчислення площі кільця
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	float rl,r2;		// радіус кільця і отвору
	float s;		// площа кільця
	printf("\nВведіть данні:\n");
	printf("радіус кільця (см) -> ");
	scanf("%f",&rl);
	printf("радіус отвіру (см) -> ");
	scanf("%f",&r2);
	if (rl > r2)
	{
		s = 2 * 3.14 * (rl - r2);
		printf("\nПлоща кільця %6.2f кв.см\n", s);
	}
	else
	{
		printf("\nПомилка! Радіус отвору не може бути ");
		printf("більше радіусу кільця.\n ");
	}
	return 0;
}
```

5) Написати програму вирішення квадратного рівняння. програма повинна перевіряти правильність вихідних даних і в випадку, якщо коефіцієнт при другому ступені невідомого дорівнює нулю, виводити відповідне повідомлення. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
    Рішення квадратного рівняння
    Введіть значення коефіцієнтів и натисніть <Enter»
    -> 12 27 -10
    Корні рівняння:
```

```
x1 = -25.551
x2 = -28.449
```

Вирішення задачі 5:

```
// Рішення квадратного рівняння
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	float a, b, c;		// коефіцієнти рівняння
	float xl, x2;		// корні рівняння
	float d;			// дискримінант
	printf("\nРішення квадратного рівняння\n");
	printf("Введіть значення коефіцієнтів и натисніть ");

	printf("<Enter»\n");
	printf("-> ");
	scanf("%f%f%f", &a, &b, &c); // введення коефіцієнтів
	d = b * b - 4 * a * c;		// дискримінант
	if (d < 0) {
		printf("Рівняння не має рішення\n");
	}
	else
	{
		xl = (-b + sqrt(d)) / (2 * a);
		x2 = (-b - sqrt(d)) / (2 * a);
		printf("Корні рівняння: xl=%3.2f x2=%3.2f\n", xl,
x2);
	}
	return 0;
}
```

6) Написати програму, яка виводить приклад на множення двох однозначних чисел, запитує відповідь користувача, перевіряє його і виводить повідомлення «Правильно!» або «Ви помилилися» і правильний результат. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Перевірка вміння множити числа
Скільки буде 6 х 7 ?
```

```
Введіть відповідь и натисніть <Enter>
-> 56
Ви помилились -> 6 x 7 = 42
```

Вирішення задачі 6:

```
// Перевірка вміння множити числа
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
#include <stdlib.h>          //для доступу к srand
#include <time.h>            //для доступу к time
using namespace std;

int main()
{
       SetConsoleCP(1251);
       SetConsoleOutputCP(1251);
       int ml, m2, p;        // співмножники і добуток
       int otv;              // відповідь
       time_t t;             // поточний час - для ініціалізації

                             // генератора випадкових чисел

       srand((unsigned) time(&t)); // ініціалізація генератора
                                   // випадкових чисел
       ml = rand() % 9 +1;   // залишок від ділення rand() на 9
                             // лежить в діапазоні від 0 до 8
       m2 = rand() % 9 +1;
       p = ml * m2;
       printf("Скільки буде %i x %i ?\n", ml, m2);
       printf("Введіть відповідь и натисніть <Enter>\n");
       printf("-> ");
       scanf("%i", &otv);
       if (p == otv)
              printf("Вірно");
       else
              printf("Ви помилились -> %i x %i = %i", ml, m2, p);
       return 0;
}
```

7) Написати програму, яка запитує у користувача номер місяця і потім виводить відповідну назву часу року. Якщо користувач введе неприпустиме число, програма повинна вивести повідомлення «Помилка даних».

Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Визначення пори року за номером місяця
Введіть номер місяця (число від 1 до 12)
-> 11
Зима
```

Вирішення задачі 7:

```
// Визначення пори року за номером місяця
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main() {
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int month;		// номер місяця
	puts("\n Визначення пори року за номером місяця");
	puts("\n Введіть номер місяця (число від 1 до 12)\n");
	printf("-> ");
	scanf("%i", &month);
	if (month < 1 && month > 12)
		printf("Число повинно бути від 1 до 12");
	else
		if (month >= 3 && month <= 5)
			printf("Весна");
		else
			if (month >= 6 && month <= 8)
				printf("Літо");
			else
				if (month >= 9 && month <= 11)
					printf("Осінь");
				else
					printf("Зима");
	return 0;
}
```

8) Написати програму, яка обчислює дату наступного дня. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Обчислення дати наступного дня
Введіть цифрами сьогоднішню дату
(число місяць рік) -> 31 12 2019
Останній день місяця!
З наступаючим Новим роком!
Завтра 1.1.2020
```

Вирішення задачі 8:

```
// Обчислення дати наступного дня
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int day;
	int month;
	int year;
	int last;	// 1, якщо поточний день останній день місяця
	int r;		// якщо рік високосний, то залишок від
			// ділення year на 4 дорівнює нулю
	printf("Введіть цифрами сьогоднішню дату\n");
	printf("(число місяць рік) -> ");
	scanf("%i%i%i", &day, &month, &year);
	last = 0;
	if (month == 2)
	{
		if ((year % 4) != 0 && day == 28) last = 1;
		if ((year % 4) == 0 && day == 29) last = 1;
	}
	else
		if ((month == 4 || month == 6 ||
		     month == 9 || month == 11) && (day == 31))
			last = 1;

		else
			if (day == 31)
				last = 1;
	if (last == 1)
	{
```

```
            printf("Останній день місяця!\n");
            day = 1;
            if (month == 12)
            {
                    month = 1;
                    year++;
                    printf("З наступаючим Новим роком!\n");
            }
            else
                     month++;
        }
        else
        day++;

        printf ("Завтра %i %i %i", day, month, year);
        return 0;
}
```

9) Програма перекладу літерних оцінок в цифрові. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Виводить цифрову оцінку
Введіть оцінку у вигляді літери (A .. E) -> A
Ваша оцінка – 5
```

Вирішення задачі 9:

```
// Виводить цифрову оцінку
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        char oc;              // оцінка у вигляді літери
        puts("\n Виводить цифрову оцінку\n");
        puts("\nВведіть оцінку у вигляді літери (A .. E) ");
        printf(" ->");
        scanf("%c", &oc);
        puts("\nВаша оцінка –");
```

```
        switch (oc)
        {
                case 'A': puts(" 5 "); break;
                case 'B': puts(" 4 "); break;
                case 'C': puts(" 3 "); break;
                case 'D': puts(" 2 "); break;
                case 'E': puts(" 1 "); break;
                default:
                puts("Оцінка повинна бути в діапазоні A .. E\n");
        }
        return 0;
}
```

10) Написати програму, яка за датою визначає відповідний їй день тижня. Для обчислення дня тижня треба скористатися формулою:

$$\left(d + \left[\frac{1}{5}(13m - 1)\right] + Y + \left[\frac{Y}{4}\right] + \left[\frac{c}{4}\right] - 2c + 777\right) mod 7\ ,$$

де d – число місяця, m – номер місяця, якщо починати рахунок з березня, як це робили в Стародавньому Римі (березень –1, квітень – 2, ..., лютий – 12), Y – номер року в столітті, c – кількість століть. *Квадратні дужки* означають, що потрібно взяти цілу частину від укладеного в них значення. Обчислене за формулою число визначає день тижня: 1 – понеділок, 2 – вівторок, ..., 6 – субота, 0 – неділя. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Визначення дня тижня за датою
Введіть дату: день місяць рік. Наприклад, 5 12 2019
-> 8 3 2020
Неділя
```

Вирішення задачі 10:

```
// Визначення дня тижня за датою
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;
```

```
int main() {
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int day, month, year; // день, місяць, рік
	int c, y;             // століття и рік в столітті
	int m;        // місяць по давньоримському календарю
	int d;                // день тижня

	puts("\nВизначення дня тижня за датою\n");
	puts("Введіть дату: день місяць рік.");
	puts("Наприклад, 5 12 2020\n ");
	printf("->");
	scanf("%i %i %i", &day, &month, &year);

	if (month == 1 || month == 2)
		year--;       // січень та лютий відносяться
				// до попереднього року

	m = month - 2;        // рік починається з березня
	if (m <= 0)
		m += 12;      // для січня та лютого
				// m - номер місяцч по римському
				//календарю
	c = year / 100;
	y = year - c * 100;
	d = (day + (13 * m - 1) / 5 + y + y / 4 + c / 4 - 2 * c +
777) % 7;
	switch (d) {
	case 1: puts("Понеділок"); break;
	case 2: puts("Вівторок"); break;
	case 3: puts("Середа"); break;
	case 4: puts("Четвер"); break;
	case 5: puts("П'ятниця"); break;
	case 6: puts("Субота"); break;
	case 0: puts("Неділя"); break;
	}
	return 0;
}
```

11) Програма найпростішого калькулятору. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Найпростіший калькулятор
```

```
Введіть арифметичний вираз (у вигляді – число знак число) ->
5 + 3
= 8
```

Вирішення задачі 11:

```
// Найпростіший калькулятор
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int a, b;		//введені з клавіатури числа
	char c;			//введений з клавіатури знак операції
	puts("\n Найпростіший калькулятор \n");
	puts("\n Введіть арифметичний вираз (у вигляді –");
	puts(" число знак число) ");
	printf(" ->\n ");
	scanf("%i %c %i", &a, &c, &b);
	switch (c)
	{
		case '+':
			printf(" = %i", a + b); break;
		case '-':
			printf(" = %i", a - b); break;
		case '*':
			printf(" = %i", a * b); break;
		case '/':
			if (b == 0)
				printf("Ділити на нуль не можна!");
			else
				printf(" = %f", (float)a / b);
			break;
		default:
			puts("Невірний знак операції\n");
	}
	return 0;
}
```

Практична робота 5

ЦИКЛИ. ОПЕРАТОРИ ПЕРЕХОДУ

Мета: Оволодіння практичними навичками роботи з операторами циклу та операторами переходу в мовах C/C++

Теми для попереднього опрацьовування

- Логічні операції
- Умовні оператори

Теоретичні відомості

Цикли служать для виконання деякого фрагмента програми кілька разів. У окремих випадках фрагмент виконується в кожному послідовному кроці циклу без змін. Частіше в кожному кроці циклу виконання фрагменту деяк відрізняється від попереднього його виконання. Цикл може виконуватися впродовж заздалегідь заданого числа кроків, а може завершуватися при виконанні деякої умови.

Існує три види циклів в мові C: *while*, *for* і *do*.

Оператор циклу з передумовою

Цикл ***while*** має наступну форму:

```
while (вираз) тіло циклу;
```

Тіло циклу виконується до тих пір, поки значення виразу дорівнює *true*. Вираз обчислюється перед кожним виконанням тіла циклу. Як тільки вираз прийме значення *false*, відбудеться вихід з циклу. Значення виразу має змінюватись в результаті виконання деяких дій в тілі циклу. Оператори в тілі циклу можуть зовсім не виконуватись, якщо вираз одразу має значення *false*. Тіло циклу може складатись з одного або декількох операторів.

Приклад 1. Програма підраховує символи у вхідному потоці. Вихід з циклу – при введенні символу ‘*’.

```
#include <stdio.h>
void main()
{
   int k = 0;
   char c;
   while| ((c|із| = getchar|()) != '*'|)
       k| += 1;
    printf| ("%d characters| \n", k|);
  }
```

Оператор циклу for

Цикл ***for*** має наступну форму:

```
for (вираз1; вираз2; вираз3) тіло циклу;
```

Цикл *for* є зручним скороченим записом для циклу *while* вигляду

```
вираз1;
while (вираз2)
{
   тіло циклу;
   вираз3;
}
```

В циклі *for* вираз1 служить для завдання початкових умов виконання циклу, вираз2 забезпечує перевірку умови виходу з циклу, а вираз3 модифікує умови, задані виразом1. Будь-який з виразів може бути відсутнім. Якщо відсутній вираз2, то за умовчанням замість нього підставляється значення *true*. Наприклад, в операторі *for*

```
for (;вираз2; ) тіло циклу;
```

відсутні вираз1 і вираз3**.** Цей оператор еквівалентний циклу

```
while (вираз2) тіло циклу;
```

Оператор циклу *for*

```
for (;;) тіло циклу;
```

зі всіма відсутніми виразами еквівалентний циклу

```
while (true) тіло циклу;
```

тобто він еквівалентний нескінченному циклу. Такий цикл може бути перерваний тільки явним виходом з нього за допомогою операторів *break*, *goto* або *return*, якщо якийсь з них міститься в тілі циклу.

Приклад 2. Підрахувати суму десяти чисел, введених з клавіатури.

```
#include <stdio.h>
void main()
{
  int sum = 0, k, i;

  for (i=0; i<10; i++)
  {
     scanf("%d",&k);
     sum +=k;
   }
  printf| ("%d summa| \n", sum|);
}
```

Оператор циклу з постумовою
Цикл ***do*** має наступну форму:

```
do тіло циклу while (вираз);
```

Тіло циклу виконується до тих пір, поки вираз має значення *true*. На відміну від циклу *while*, в якому перевірка умови закінчення циклу робиться до виконання тіла циклу, в циклі *do* така перевірка має місце після виконання тіла циклу. Отже, тіло циклу *do* буде виконано хоч би один раз, навіть якщо вираз має значення *false* із самого початку.

Приклад 3. Підрахувати суму чисел, введених з клавіатури, поки не буде введене число нуль.

```
#include <stdio.h>
void main()
{
  int s = 0, a;
  do
  {
     scanf("%d",&a);
     s+=a;
  }
  while(a!=0);
      printf| ("%d summa| \n", s|);
  }
```

Інші управляючі оператори

Оператор ***break.***

Оператор *break* використовується для виходу з операторів *while*, *do*, *for*, *switch*, якщо ці оператори безпосередньо його містять. Управління передається до оператора, наступного за оператором, з якого здійснений вихід. Оператор *break* має форму

```
break;
```

Приклад 4. З клавіатури вводиться символ *ch*. Якщо він не співпадає з символом ‘*’, виконується тіло оператора *while.* Робота циклу повністю припиняється, як тільки при введенні зустрічається символ “новий рядок” (‘*n*’).

```
 while((ch=getchar()) != '*')
{
     if(ch=='\n')break;
         putchar(ch);
}
```

Оператор ***continue***

Оператор *continue* служить для пропуску частини виконуваної ітерації циклу, що безпосередньо його містить Якщо умовами циклу допускається нова ітерація, то вона виконується, інакше цикл завершується. Оператор *continue* має наступну форму:

```
continue;
```

Приклад 5. З клавіатури вводиться символ *ch.* Якщо він не співпадає з символом '*', виконується тіло оператора *while*. У версії з оператором *continue* просто пропускаються символи "новий рядок" ('\n'), тобто не виконується перехід на новий рядок при виведенні на екран, а вихід з циклу виконується, тільки коли вводиться символ '*'.

```
while((ch=getchar()) != '*')
{
   if(ch=='\n')  continue;
   putchar(ch);
}
```

Оператор ***goto***

Оператор *goto* призначений для безумовної передачі управління оператору з вказаною міткою. Він має наступну форму:

```
goto мітка;
```

Наприклад, використання оператора *goto part1;* передбачає наявність іншого оператора, що має мітку *part1*. В цьому випадку запис іншого оператора починається з мітки, за якою слідує двокрапка:

```
part1: printf("точка переходу\n");
```

Задачі

Загальні умови:

Кожне завдання містить приклад використання операцій циклу мов C/C++, і може містити умови вирішення. Необхідно з'ясувати, що буде отримано програмою, яка реалізує запропоновані операції. Після списку завдань надано рішення у вигляді програмного коду.

1) Обчислити суму квадратів натурального ряду чисел від 1 до *n*, використовуючи оператор циклу *for*

$$S = \Sigma (1^2 + 2^2 + \ldots + n^2).$$

Вирішення задачі 1:

```
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int S, n, i;
	printf("Введіть n :");
	scanf(" %d", &n);
	for (S = 0, i = 1; i <= n; i++)
	{
		S += i * i;	//	відповідає S = S + i * i
	}
	printf("n = %d S = %d", n, S);
	return 0;
}
```

2) Розглянути приклад з попередньої роботи. Визначити число *n*, при якому сума чисел *S* попереднього ряду не перевищить величину *K*, яку введено з клавіатури. Програму реалізувати за допомогою циклів з *передумовою* і з *постумовою*.

Вирішення задачі 2:

```
// цикли while та do…while
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int S, K, i;
	printf("Введіть K :");
	scanf(" %d", &K);
	S = 0;
	i = 0;
	while (S < K)		// цикл с передумовою
	{
		i++;
		S += i * i;
	}
	printf("n = %d S = %d \n", i, S);
	S = 0;
	i = 0;
	do			// цикл с постумовою
	{
		i++;
		S += i * i;
	} while (S < K);
	printf("n = %d S = %d \n", i, S);
	return 0;
}
```

3) Написати програму, яка виводить таблицю квадратів перших п'яти цілих позитивних чисел. Нижче наведено рекомендований вид екрану програми:

```
Таблиця квадратів
-----------------
Число    Квадрат
1  1
2  4
3  9
4  16
5  25
```

Вирішення задачі 3:

```
// Виводить таблицю квадратів перших п'яти цілих
// позитивних чисел
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int x = 1;		// перше число
	int y;			// квадрат числа
	int i;			// лічильник циклів
	printf("Таблиця квадратів\n");
	printf("................ \n");
	printf("Число   Квадрат\n");
	printf("................ \n");
		for (i = 1; i <= 5; i++)
		{
			y = x * x;
			printf("%3i\t%4i\n", x, y);
			x += 1;
		}
	printf("................\n");
}
```

4) Написати програму, яка вводить з клавіатури послідовність з п'яти дрібних чисел і після введення кожного числа виводить середнє арифметичне введеної частини послідовності. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Обробка послідовності дрібних чисел
Після введення кожного числа натисніть <Enter>
-> 12.3
Введено чисел: 1 Сума: 12.30 Середнє арифметичне:
12.30
-> 15
```

```
Введено чисел: 2 Сума: 27.30 Середнє арифметичне:
13.65
-> 10
Введено чисел: 3 Сума: 37.30 Середнє арифметичне:
12.43
-> 5.6
Введено чисел: 4 Сума: 42.90 Середнє арифметичне:
10.73
-> 11.5
Введено чисел: 5 Сума: 54.40 Середнє арифметичне:
10.88
```

Вирішення задачі 4:

```
// Середнє арифметичне дрібних чисел,
// які вводяться з клавіатури
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;
#define L 5          // кількість чисел послідовності

int main() {
       SetConsoleCP(1251);
       SetConsoleOutputCP(1251);

       float a;      // число
       int n;        // кількість введених чисел
       float sum;    // сума введених чисел
       float sred;   // середнє арифметичне введених чисел

       printf("\nОбробка послідовності дрібних чисел\n");
       printf("Після введення кожного числа ");
       printf("натисніть < Enter > \n");
       sum = 0;
       for (n = 1; n <= L; n++)
       {
              printf("-> ");
              scanf("%f", &a);
              sum += a;
              printf("Введено чисел : % i ", n);
              printf("Сумма: %6.2f\n", sum);
              sred = sum / n;
              printf("Середнє арифметичне: %6.2f\n", sred);
```

```
        }
        return 0;
}
```

5) Написати програму, яка перетворює введене користувачем десяткове число в двійкове. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Перетворення десяткового числа в двійкове
Введіть ціле число від 0 до 255 і натисніть <Enter>
-> 49
Десятковому числу 49 відповідає двійкове 00110001
```

Вирішення задачі 5:

```
// Перетворення десяткового числа в двійкове
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int dec;              // десяткове число
        int v;                // вага формованого розряду
        int i;                // номер формованого розряду
        printf("\Перетворення десяткового числа в двійкове \n");
        printf("Введіть ціле число від 0 до 255 ");
        printf("і натисніть < Enter> -> ");
        scanf("%i", &dec);
        printf("Десятковому числу %i відповідає двійкове ", dec);
        v = 128;              // вага старшого (восьмого) розряду
        for (i = 1; i <= 8; i++)
        {
                if (dec >= v)
                {
                        printf("1");
                        dec -= v;
                }
                else
                        printf("0");
                v = v / 2;            // вага наступного розряду
                                      // в два рази менше
```

```
        }
        return 0;
}
```

6) Написати програму, яка обчислює суму і середнє арифметичне послідовності позитивних чисел, які вводяться з клавіатури. Для завершення введення ввести нуль. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Обчислення середнього арифметичного
послідовності позитивних чисел.
Введіть числа. Для завершення введення введіть нуль.
-> 45
-> 23
-> 15
-> 0
Введено чисел: 3
Сума чисел: 83
Середнє арифметичне: 27.67
```

Вирішення задачі 6:

```
// Обчислення середнього арифметичного послідовності позитивних
чисел
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int a;          // число, введене з клавіатури
        int n;          // кількість чисел
        int s;          // сума чисел
        float m;        // середнє арифметичне

        s = 0;
        n = 0;
        printf("\nОбчислення середнього арифметичного\n ");
        printf("послідовності позитивних чисел.\n");
```

```
	printf("Введіть числа. Для завершення введення ");
	printf("введіть	нуль.\n");
	do
	{
		printf("-> ");
		scanf("%i", &a);
		if (a > 0) {
			s += a;
			n++;
		}
	} while (a > 0);
	printf("Введено чисел: %i\n", n);
	printf("Сума чисел: %i\n", s);
	m = (float)s / n;
	printf("Середнє арифметичне: %3.2f", m);
	return 0;
}
```

7) Написати програму, яка визначає максимальне число з введеної з клавіатури послідовності позитивних чисел (довжина послідовності не обмежена). Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Визначення максимального числа з послідовності позитивних чисел.
Введіть числа. Для завершення введення введіть нуль.
-> 56
-> 75
-> 43
-> 0
Максимальне число: 75
```

Вирішення задачі 7:

```
// Визначення максимального числа з послідовності позитивних чисел
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
```

```
        SetConsoleOutputCP(1251);
        int a;          // введене число
        int m;          // максимальне число
        puts ("\nВизначення максимального числа з ");
        puts("послідовності позитивних чисел.\n");
        puts("Введіть числа. Для завершення введення введіть нуль.\n");
        m = 0;
        do
        {
                printf("-> ");
                scanf("%i", &a);
                        if (a > m)
                        m = a;
        } while (a > 0);
        printf("Максимальне число: %i", m);
        return 0;
}
```

8) Написати програму, яка перевіряє, чи є введене користувачем ціле число простим (простим називається число, яке ділиться тільки на одиницю і на саме себе). Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Введіть ціле число і натисніть <Enter>
-> 17
17 - просте число
```

Вирішення задачі 8:

```
// Визначення простого числа
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int n;          // число
        int d;          // дільник
```

```
        int r;          // залишок від ділення n на d
        printf("Введіть ціле число і натисніть <Enter>\n-> ");
        scanf("%i", &n);
        d = 2;          // спочатку ділимо на два
        do
        {
                r = n % d;
                if (r != 0)
                        d++;
        } while (r != 0);     // поки n не поділиться на d
        if (d == n)
        {
                printf("%i - просте число", n);
        }
        else
        {
                printf("%i - не просте число", n);
        }
        return 0;
}
```

9) Написати програму, яка обчислює найбільший спільний дільник двох цілих чисел. Далі наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Визначення найбільшого спільного дільника двох цілих
чисел
Введіть два числа і натисніть <Enter>
-> 12 18
НСД чисел 12 і 18 - це 6
```

Вирішення задачі 9:

```
// Визначення найбільшого спільного дільника двох цілих чисел
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;
```

```
int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	int n1, n2;	// числа, НСД яких треба обчислить
	int nod;	// найбільший спільний дільник
	int r;		// залишок від ділення n1 на n2
	printf("\nВизначення найбільшого спільного дільника ");
	printf("двох цілих чисел.\n");
	printf("Введіть два числа і натисніть <Enter>\n");
	printf("-> ");
	scanf("%i %i", &n1, &n2);
	printf("НСД чисел %i і %i - це ", n1, n2);
	while (n1 % n2)
	{
		r = n1 % n2; // залишок від ділення
		n1 = n2;
		n2 = r;
	}
	nod = n2;
	printf("%i\n", nod);
	return 0;
}
```

10) Роздрукувати літери латинського алфавіту відповідно до наведеного нижче видом:

```
A B C D E
B C D E F
C D E F G
D E F G H
E F G H I
```

Вирішення задачі 10:

```
// Друк букв латинського алфавіту
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
```

```
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int i, j, c;
	// Зовнішній цикл забезпечує виведення 5 рядків
	// з початковими символами 'A', 'B', 'C', 'D', і 'E'
	for (i = 'A'; i <= 'E'; i++)
	{
	// Внутрішній цикл забезпечує виведення рівно 5 символів
	//починаючи з символу i
		for (c = i, j = 0; j < 5; c++, ++j)
		{
			printf("%c   ", c);
		}
		printf("\n");
	}
	return 0;
}
```

Практична робота 6

РОБОТА З МАСИВАМИ

Мета: Оволодіння практичними навичками роботи з одномірними масивами в мовах С/С++

Теми для попереднього опрацьовування

– Оператори циклу мов С/С++. Вкладені цикли
– Умовні оператори мов С/С++
– Оператори введення/виведення мови С++

Теоретичні відомості

Масив – це впорядкований набір об'єктів, у яких однаковий тип, наприклад, набір цілих чисел або символів. У масиві можна зберігати якусь кількість однотипних об'єктів без необхідності введення окремого імені змінної для кожного з них. Масив може бути одновимірним або багатовимірним. Кількість вимірювань і довжина кожного з вимірювань визначає загальну довжину масиву, яка обмежена лише адресним простором конкретного комп'ютера.

Масив, як і будь-який інший об'єкт в мовах С/С++, має бути оголошений перед тим, як він буде використаний:

```
тип ідентифікатор [довжина 1]...[довжина N];
```

У оголошенні масиву в квадратних дужках після *ідентифікатора* (імені масиву) вказується *довжина* кожного вимірювання. Кількість вимірювань масиву визначається по числу квадратних дужок в його оголошенні.

Індекс масиву визначає елемент масиву, до якого треба забезпечити доступ, і вказується в квадратних дужках після імені масиву. Індекс масиву – це цілочисельний вираз, значення якого може бути в діапазоні від 0 до значення, рівного довжині вимірювання, зменшеній на 1. Наприклад:

```
char id[4];
```

Це оголошення відводить всього 4 байта для 4-елементного масиву символів. Елементи масиву *id* це – *id* [0], *id* [1], *id* [2], *id* [3].

```
int table[3][3];
```

Це є оголошення двовимірного масиву цілих чисел. Цей масив можна розглядати як набір з трьох 3-елементних масивів цілих чисел.

Аналогічно, тривимірний масив *val*[5][4][3] можна розглядати як п'ять масивів розмірності 4 на 3.

Елементи масиву зберігаються в пам'яті послідовно. Якщо, наприклад, перший елемент символьного масиву *id* зберігається за адресою 5000, то другий зберігатиметься за адресою 5001, третій – за адресою 5002.

Двовимірний масив розташовується в пам'яті за рядками: спочатку – нульовий рядок, потім – перший і так далі Можна сказати, що "наступний індекс міняється швидше за попередній". Так, елементи *table*[3][3] зберігаються в наступному порядку: [0][0], [0][1], [0][2], [1][0], [1][1], [1][2], [2][0], [2][1], [2][2].

Операція доступу до елемента масиву – операція "квадратні дужки", яка використовується таким чином: *ім'я_массиву*[*індекс*].

Ця операція відноситься до первинних і знаходиться в найвищому рядку таблиці пріоритетів операцій мов C/C++.

Ім'я_массиву – ідентифікатор, якому ставиться у відповідність початкова адреса масиву (адреса першого елемента).

У дужках повинен стояти цілочисельний вираз, значення якого має знаходитися в діапазоні від нуля до (*розмір масиву* – 1). Наприклад:

```
char id[3];
id[0] = 'j';
```

Елементу з номером нуль масиву *id* привласнюється код символу '*j*'.

Якщо в масиві не дуже багато елементів і їх значення відомі заздалегідь, масив можна ініціалізувати разом з його оголошенням, розмістивши перелік значень у фігурних дужках:

```
int iNs[10] = {0,1,2,3,4,5,6,7,8,9};
```

Якщо ж в масиві багато елементів, заповнити його їх значеннями доведеться програмно в циклі.

Приклад 1. Утворити масив зі ста елементів, заповнених натуральним рядом чисел.

```
int imas[100], i;
for (i = 0; i < 100; i++)
   imas[i] = i*i;
```

Копіювання масивів

Масиви треба копіювати поелементно.

Приклад 2. Копіювання масивів.

```
int prev[20], current[20], i;
i = 0;
while (i < 20)
{
   prev[i]= current[i];
   i++;
}
```

Приклад 3. Підрахувати найбільше число однакових елементів масиву, які розміщені підряд. Умовна операція (*k* > *max*) ? *k* : *max* в операторі *printf* передбачена для того окремого випадку, коли весь масив складається з однакових елементів.

```
#include <stdio.h>
void main()
{
   int i, k, max, n = 10;
   int a[] = {5, 6, 6, 6, 4, 3, 3,3, 3, 3, 8};
   i = k = max = 1;
   while(i < n)
   {
     if(a[i] == a[i-1])
          k++;
     else
     {
```

```
            if(k > max)
                    max = k;
            k = 1;
        }
        i++;
    }
    printf("kmax = %d\n",(k > max)? k : max);
}
```

Оператор ***cout***

Оператор *cout* (який знаходиться в бібліотеці *iostream*) використовується для виводу даних на екран (у консольному вікні). Наприклад:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, world!";
    return 0;
}
```

Для виводу декількох виразів на одній строчці оператора виводу << потрібно використовувати кілька разів, наприклад:

```
#include <iostream>
using namespace std;
int main()
{
   int a = 7;
   cout << "a is " << a;
   return 0;
}
```

Оператор ***endl***

Якщо текст потрібно вивести окремо (на декількох рядках) – треба використовувати *endl*. При використанні з *cout*, *endl* вставляє символ нового рядка. Таким чином, текст переноситься до початку наступного рядка, наприклад:

```
#include <iostream>
```

```
using namespace std;
int main()
{
   cout << "Hi!" << std::endl;
   cout << "My name is Anton." << std::endl;
   return 0;
}
```

Оператор ***cin***

Оператор *cin* є протилежністю *cout*. У той час як *cout* виводить дані в консоль за допомогою оператора виведення <<, *cin* отримує дані від користувача за допомогою оператора введення >>. Використовуючи *cin* можемо отримувати і обробляти введення користувача:

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Enter a number: ";
   int a = 0;
   cin >> a;
   cout << "You entered " << a << endl;
   return 0;
}
```

Задачі

Загальні умови:

Кожне завдання містить приклад використання операцій роботи з одновимірними масивами мов C/C++, і може містити умови вирішення. Необхідно з'ясувати, що буде отримано програмою, яка реалізує запропоновані операції. Після списку завдань надано рішення у вигляді програмного коду.

1) Написати програму, яка записує введені з клавіатури дані в одновимірний масив цілого типу, що складається з семи елементів. Перед введенням кожного елемента повинна виводитися підказка з номером елемента. Після введення останнього елемента програма повинна вивести введений масив і обчислити середнє арифметичне його елементів. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем,

виділені напівжирним):

```
Введення масиву цілих чисел
Після введення кожного числа натисніть <Enter>
a[0] -> 10
a[1] -> 16
a[2] -> 14
a[3] -> 5
a[4] -> 10
a[5] -> 22
a[6] -> 22
Масив: 10 16 14 5 10 22 22
Середнє арифметичне: 14.00
```

Вирішення задачі 1:

```
// Введення та обробка масиву
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int a[7];		// масив
	int sum;		// сума елементів масиву
	float m;		// середнє арифметичне
	printf("\n Введення масиву цілих чисел ");
	printf("\n Після введення кожного числа натисніть
Enter>\n");
	// введення масиву
	for (int j = 0; j < 7; j++)
	{
		printf("\na[%i] -> ", j);
		scanf("%i", &a[j]);
	}
	// виведення масиву
	printf("\nМасив: \n");
		for (int j = 0; j < 7; j++)  {
			printf("%i ", a[j]);
		}
	sum = 0;
```

```
        // обчислення суми елементів
        for (int j = 0; j < 7; j++)
        {
                sum = sum + a[j];
        }
        m = sum / 7;
        printf("Середнє арифметичне: %f", m);
        return 0;
}
```

2) Написати програму, яка перевіряє, чи знаходиться введене з клавіатури число в масиві. Масив повинен вводитися в процесі роботи програми. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Пошук в масиві методом перебору
Введіть однією строчкою 5 цілих чисел и натисніть <Enter>
16 14 5 10 22
Введіть зразок для пошуку (ціле число)-> 10
Збіг з елементом номеру 3
```

Вирішення задачі 2:

```
// Пошук в масиві методом перебору елементів
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

#define HB 5

int main() {
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int m[HB];            // масив цілих чисел
        int obr;              // зразок для пошуку
        int found;            // ознака збігу зі зразком
        printf("\n Пошук в масиві методом перебору \n");
        printf("Введіть однією строкою %i цілих ", HB);
        printf("чисел и натисніть <Enter>\n");
        printf("->");
        for (int i = 0; i < HB; i++)
                scanf("%i", &m[i]);
```

```
        printf("Введіть зразок для пошуку (ціле число) -> ");
        scanf("%i", &obr);
        // пошук перебором
        found = 0;
        int i = 0;     // перевірка з першого елемента масиву
        do  {
               if (m[i] == obr)
                      found = 1; //збіг зі зразком
               else
                      i++; // перехід до наступного елементу

        } while (!found && i < НВ);
        if (found)
               printf("Збіг з елементом номеру %i", i + 1);
        else
               printf("Збігу зі зразком нема");
        return 0;
}
```

3) Написати програму, яка перевіряє, чи становлять елементи введеного з клавіатури масиву зростаючу послідовність. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Перевірка, чи відсортований чи масив по зростанню
Введіть масив з 5 цілих чисел и натисніть <Enter>
16 14 5 10 22
Елементи масиву не упорядковані по зростанню
```

Вирішення задачі 3:

```
// Перевірка, чи відсортований чи масив по зростанню
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

#define НВ 5

int main() {
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int a[НВ];           // масив цілих чисел
```

```
        int ok;              // 1 - послідовність неубутна
        int k;               //індекс
        printf("\nПеревірка, чи відсортований чи масив ");
        printf("по зростанню \n");
        printf("Введіть масив з %i цілих чисел и натисніть", НВ);
        printf(" <Enter>\n");
        for (int k = 0; k < НВ; k++)
               scanf("%i", &a[k]);
        k = 0;
        ok = 1;
        do {
               if (a[k] > a[k + 1])
                      ok = 0;
               k++;
        } while (k < НВ - 1 && ok);
        printf("\nЕлементи масиву ");
        if (!ok)
               printf("не ");
        printf("упорядковані по зростанню");
        return 0;
}
```

4) Написати програму, яка методом обміну («бульбашки») сортує по зростанню введений з клавіатури одновимірний масив. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Сортування масиву методом "бульбашки"
Введіть масив з 5 цілих чисел и натисніть <Enter>
16 14 5 10 22
Сортування ...
Масив відсортований -> 5 10 14 16 22
```

Вирішення задачі 4:

```
// Сортування масиву методом "бульбашки"
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;
```

```
#define SZ 5

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int a[SZ];
	int i;		// лічильник циклів
	int k;		// поточний індекс елемента масиву
	int buf;
	printf("\nСортування масиву методом \" бульбашки \"\n");
	printf("Введіть масив з %i цілих чисел и натисніть", SZ);
	printf(" <Enter>\n");
	for (k = 0; k < SZ; k++)
		scanf("%d", &a[k]);
	printf("Сортування...\n");
	for (i = 0; i < SZ - 1; i++)
	{
		for (k = 0; k < SZ - 1; k++)
		{
			if (a[k] < a[k + 1])
			{// обмін k-го і (k + 1)-го елементів
				buf = a[k];
				a[k] = a[k + 1];
				a[k + 1] = buf;
			}
		}
	}
	printf("Масив відсортований -> ");
	for (k = 0; k < SZ; k++)
		printf("%d ", a[k]);
	return 0;
}
```

5) Написати програму, яка об'єднує два упорядкованих по зростанню масиву в один, який також повинен бути впорядкований по зростанню. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Об'єднання двох упорядкованих за зростанням масивів
Введіть перший масив (5 цілих чисел) -> 1 3 5 7 9
Введіть другий масив (5 цілих чисел) -> 2 4 6 8 10
Масив - результат
1 2 3 4 5 6 7 8 9 10
```

Вирішення задачі 5:

```
// Об'єднання двох упорядкованих за зростанням масивів
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

#define SZ 5

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int a[SZ], b[SZ];	// вхідні масиви
	int c[SZ * 2];		// масив - результат
	int k, i, m;		// індекси масивів a, b і c
	printf("Об'єднання двох упорядкованих ");
	printf("за зростанням масивів \n");
	printf("Введіть перший масив ");
	printf("(5 цілих чисел) ->  ", SZ);
	for (k = 0; k < SZ; k++)
		scanf("%i", &a[k]);
	printf("Введіть другий масив ");
	printf("(5 цілих чисел) ->  ", SZ);
	for (i = 0; i < SZ; i++)
		scanf("%i", &b[i]);
	k = i = m = 0;
	do
	{
		if (a[k] < b[i])
			c[m++] = a[k++];
		else
			if (a[k] > b[i])
				c[m++] = b[i++];
			else
			{
				c[m++] = a[k++];
```

```
                        c[m++] = b[i++];
                }
        } while (k < SZ && i < SZ); // поки один з двох вихідних
                                    // масивів повністю не
                                    // переписаний в масив с
        while (k < SZ)// є елементи a, що не переписані в с
                c[m++] = a[k++];
        while (i < SZ) // є елементи b, що не переписані в с
                c[m++] = b[i++];
        printf("Масив - результат: \n");
        for (i = 0; i < 2 * SZ; i++)
                printf("%i ", c[i]);
        return 0;
}
```

6) Написати програму, яка в одновимірному масиві парні елементи замінює на нуль та рахує кількість непарних елементів. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Заміна парних елементів та кількість непарних
Введіть масив (5 цілих чисел) -> 7 -6 14 3 8
Масив - результат  -> 7 0 0 3 0
Непарних елементів 2
```

Вирішення задачі 6:

```
// Заміна парних елементів та кількість непарних
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

#define SZ 5

int main() {
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int a[SZ];              // вхідний масив
        int count;              //кількість непарних елементів
        printf("\n Заміна парних елементів та ");
```

```
        printf("кількість непарних\n");
        printf("Введіть масив (%i цілих чисел) -> ", SZ);
        for (int k = 0; k < SZ; k++)
               scanf("%i", &a[k]);
        count = 0;
        for (int k = 0; k < SZ; k++)
        {
               if ((a[k] % 2) == 0)
                      a[k] = 0;
               else
                      count++;
        }
        printf("Масив - результат -> ");
        for (int i = 0; i < SZ; i++)
               printf("%i ", a[i]);
        printf("\nНепарних елементів %i ", count);
        return 0;
}
```

7) Написати програму, яка в одновимірному масиві для елементів з непарними індексами знаходить мінімальний і його порядковий номер. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Пошук мінімального з непарних
Введіть масив (5 цілих чисел) -> 7 -6 14 3 8
Мінімальний елемент -6
Порядковий номер 1
```

Вирішення задачі 7:

```
// Пошук мінімального з непарних та його порядкового номеру
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

#define SZ 5

int main()
{
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int a[SZ];          // вхідний масив
```

```
        int min, i_min;    //мінімальний та його порядковий номер
        printf("\n Пошук мінімального з непарних \n");
        printf("Введіть масив (%i цілих чисел -> ", SZ);
        for (int k = 0; k < SZ; k++)
               scanf("%i", &a[k]);
        min = 100;
        for (int k = 0; k < SZ; k++)
        {
               if (k % 2 != 0 && a[k] < min) {
                      min = a[k];
                      i_min = k;
               }
        }
        printf("Мінімальний елемент %i", min);
        printf("\nПорядковий номер %i\n", i_min);
        return 0;
}
```

8) Написати програму, яка формує з вхідного масиву новий масив, який складається тільки з позитивних елементів. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Масив тільки з позитивних
Введіть масив (8 цілих чисел) -> 7 -6 14 3 8 –11 –3 9
Новий масив -> 7 14 3 8 9
```

Вирішення задачі 8:

```
// Масив тільки з позитивних
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

#define SZ 8

int main()
{
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int a[SZ];            // вхідний масив
        int b[SZ];            // вихідний масив
```

```
	int i_b;		// індекс вихідного масиву
	printf("\n Масив тільки з позитивних \n");
	printf("Введіть масив (%i цілих чисел) -> ", SZ);
	for (int k = 0; k < SZ; k++)
		scanf("%i", &a[k]);
	i_b = 0;
	for (int k = 0; k < SZ; k++)
	{
		if (a[k] > 0)
			b[i_b++] = a[k];
	}
	printf("Новий масив -> ");
	for (int i = 0; i < i_b; i++)
		printf("%i ", b[i]);
	return 0;
}
```

9) Написати програму, яка видаляє із заданого масиву елемент із заданим індексом. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Видалення заданого елементу масиву
Введіть масив (10 цілих чисел) -> 0 1 2 3 4 5 6 7 8 9
Який елемент треба видалити (введіть індекс)-> 3
Новий масив -> 0 1 2 4 5 6 7 8 9
```

Вирішення задачі 9:

```
// Видалення заданого елементу масиву
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

#define N 10

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
```

```
	int a[N];			// вхідний масив
	int i_del;			// індекс елементу, що видаляється
	printf("\n Видалення заданого елементу масиву \n");
	printf("Введіть масив (%i цілих чисел)-> ", N);
	for (int i = 0; i < N; i++)
		scanf("%i", &a[i]);
	printf("Який елемент треба видалити (введіть індекс)->");
	scanf("%i", &i_del);
	for (int i = i_del; i < N - 1; i++)//перезаписати в масив
		a[i] = a[i + 1];	// всі елементи, після того,
					// який видаляємо
	printf("Новий масив -> ");
	for (int i = 0; i < N - 1; i++)
		printf("%i ", a[i]);
	return 0;
}
```

10) Написати програму, яка вставляє в заданий масив елемент на задану позицію. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Вставка заданого елементу масиву
Введіть масив (10 цілих чисел) -> 0 1 2 3 4 5 6 7 8 9
Який елемент треба вставити -> 100
На яку позицію (введіть індекс)-> 5
Новий масив -> 0 1 2 3 4 100 5 6 7 8 9
```

Вирішення задачі 10:

```
// Вставка заданого елемента в масив
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;

#define N 11

int main() {
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int a[N];			// вхідний масив
```

```
        int i_ins;            // індекс елементу, що вставляється
        int vol;              // елемент, що вставляється
        printf("\n Вставка заданого елемента в масив\n");
        printf("Введіть масив з %i цілих чисел)-> ", N - 1);
        for (int i = 0; i < N - 1; i++)
               scanf("%i", &a[i]);
        printf("Який елемент треба вставити -> ");
        scanf("%i", &vol);
        printf("На яку позицію (введіть індекс)-> ");
        scanf("%i", &i_ins);
        for (int i = N; i > i_ins; i--) // перезаписати в масив
               a[i] = a[i - 1];      // всі елементи, починаючи з
                                     // останнього, до позиції
                                     // на яку треба вставити
        a[i_ins] = vol;              //вставити на позицію i_ins
                                     //елемент vol
        printf("Новий масив -> ");
        for (int i = 0; i < N; i++)
               printf("%i ", a[i]);
        return 0;
}
```

11) Написати програму, яка виводить початкові індекси та довжину всіх позитивних послідовностей у одновимірному масиві. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним):

```
Пошук позитивних послідовностей
Введіть масив (12 цілих чисел) -> 1 3 2 -7 -4 0 -2 4 5 -6 7 8
Результат -> NP = 0  dl = 3
NP = 7  dl = 2
NP = 10  dl = 2
```

Вирішення задачі 11:

```
// Пошук позитивних послідовностей
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h"
using namespace std;
```

```
#define N 12

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int a[N];	// вхідний масив
	int fl;		// прапорець знаходження в послідовності
	int dl;		// довжина послідовності
	int NP;		// індекс послідовності
	printf("\n Пошук позитивних послідовностей \n");
	printf("Введіть масив з %i цілих чисел)-> ", N);
	for (int i = 0; i < N; i++)
		cin >> a[i];		//оператор введення
	fl = 0;				//початкові значення
	dl = 0;
	for (int i = 0; i < N - 1; i++)
	{	//умова знаходження в середині послідовності
		if (a[i] > 0 && a[i + 1] > 0 && fl)
			dl++;
		// умова знаходження на початку послідовності
		if (a[i] > 0 && a[i + 1] > 0 && !fl)
		{
			dl++;
			NP = i;
			fl = 1;
		}
		// умова виходу з послідовності
		if (a[i] > 0 && a[i + 1] <= 0 ||
		    a[i] > 0 && a[i + 1] > 0 && i + 1 == N - 1)
		{
			dl++;
			//оператор виведення
			cout << "NP = " << NP << " dl = ";
			cout << dl << endl;
			fl = 0;
			dl = 0;
		}
	}
	return 0;
}
```

Практична робота 7

ДВОВИМІРНІ МАСИВИ. РОБОТА З МАТРИЦЯМИ

Мета: Оволодіння практичними навичками роботи з двовимірними масивами мов С/С++

Теми для попереднього опрацьовування

- Оператори циклу мови С
- Вкладені цикли
- Умовний оператор мови С
- Матриці

Теоретичні відомості

Двовимірні масиви

Двовимірний масив (матриця) – це масив, де кожному елементу ставляться у відповідність два індекси. Перший індекс – номер рядка, другий індекс – номер стовпця.

При роботі з двовимірним масивом потрібно використати два оператора циклу, причому, наприклад, так: один цикл, внутрішній, потрібен для переходу між елементами рядка (тобто, по стовпчиках), а другий, зовнішній, – для переміщення між рядками.

Багатовимірні масиви, в тому числі і матриці, можна ініціювати, розглядаючи їх як масив масивів:

```
int a[3][5]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
int b[3][5]={{1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15}};
```

Матриця зберігається в пам’яті за рядками, тобто самий правий індекс в наборі індексів масиву міняється найшвидше.

Якщо в матриці кількість рядків і стовпчиків однакова, то таку матрицю називають *квадратною*. Тільки в квадратних матрицях існують головна та бічна діагоналі.

Елементи, що стоять на головній діагоналі, мають індекси: (1, 1), (2, 2), (3, 3), ..., (*i*, *i*), ..., (*n*, *n*), тобто *номер рядка і номер стовпчика однакові.* Елементи, що стоять на бічній діагоналі, мають такі індекси: (1, *n*), (2, *n* – 1), (3, *n* – 2), ..., (*i*, *n* + 1 – *i*), (*n*, 1). Тобто значення *індексів елементів розраховуються за формулою* $j = n + 1 - i$.

Дуже часто при роботі з масивами застосовується препроцесорна директива *#difine*, що дозволяє привласнювати символічні імена константам. У загальному вигляді це твердження записують так: *#define рядок*1 *рядок*2 (крапка з комою в кінці не використовується).

Перш ніж початковий текст програми буде переданий компілятору, він обробляється препроцесором, який усюди в початковому тексті замінить входження "*рядок*1" на "*рядок*2". Наприклад, директива *#difine max* 80, записана на початку програми, забезпечить усюди заміну вказаного імені *max* на константу 80.

Приклад 1. У програмі визначається мінімальний елемент кожного рядка матриці і виконується обмін місцями найденого і діагонального елементів цього ж рядка.

```
#include <stdio.h>
#define M 4
main() {
  int a[M][M]={ { 3,4,1,5),
                {-1,6,7,0},
                { 1,8,7-1},
                { 4,9,7-1}};
  int i, j, jmin, min;
  i=0;
    while(i<M) {       //знаходження мінімального елемента
      min=a[i][0];     //кожного рядка
      jmin=0;j=1;
        while(j<m){
          if(a[i][j]<min){
            min=a[i][j];
            jmin=j;
          }
          j++;
        }
      a[i][jmin]=a[i][i];  //обмін мінімального і елемента,
      a[i][i]=min;         //що стоїть на головній діагоналі
      i++;
```

```
    }
i=0;
   while(i<M){
     j=0;
        while(j<M)              //виведення масиву на екран
        printf("%3d",a[i][j++]);
        printf("\n");
     i++;
   }
}
```

Задачі

Загальні умови:

Кожне завдання містить приклад роботи в двовимірними масивами мов C/C++, і може містити умови вирішення. Необхідно з'ясувати, що буде отримано програмою, яка реалізує запропоновані операції і умови. Після списку завдань надано рішення у вигляді програмного коду.

1) Надано квадратний двовимірний масив int `A[n][n]`. Необхідно елементам, що знаходяться на головній діагоналі (ГД), що проходить з лівого верхнього кута в правий нижній (тобто тим елементам `A[i][j]`, для яких $i == j$) присвоїти значення 1, елементам, що знаходяться вище головної діагоналі ($i < j$) – значення 0, елементів, що знаходяться нижче головної діагоналі ($i > j$) – значення 2. Тобто отримати такий масив (приклад для $n == 4$):

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

Вирішення задачі 1:

```
#include <iostream>
#include "windows.h";
using namespace std;
#define n 4 // кількість рядків і стовпців

int main()
{
```

```
SetConsoleCP(1251);
SetConsoleOutputCP(1251);
int A[n][n];
// Заповнення відносно головної діагоналі
for (int i = 0; i < n; ++i)
	for (int j = 0; j < n; ++j) {
		if (i < j)
			A[i][j] = 0;
		else if (i > j)
			A[i][j] = 2;
		else
			A[i][j] = 1;
	}

// Друк матриці
printf("Спосіб 1:\n");
for (int i = 0; i < n; i++)
{
	for (int j = 0; j < n; j++) printf("%i ",
	A[i][j]);
	printf("\n");
}
printf("\n");

// Заповнення відносно побічної діагоналі (ПД)
for (int i = 0; i < n; ++i)
	for (int j = 0; j < n; ++j) {
		if (i < n - 1 - j)
			A[i][j] = 0;
		else if (i > n - 1 - j)
			A[i][j] = 2;
		else
			A[i][j] = 1;
	}

// Друк матриці
printf("Спосіб 1, але відносно побічної діагоналі:\n");
for (int i = 0; i < n; i++)
{
	for (int j = 0; j < n; j++) printf("%i ",
	A[i][j]);
	printf("\n");
}
printf("\n");
```

```
// Без використання оператору if
// Спочатку заповнення головної діагоналі
for (int i = 0; i < n; ++i)
	A[i][i] = 1;

// Потім заповнимо значенням 0 всі елементи вище головної
// діагоналі, для чого нам знадобиться в кожній з рядків
// з номером i призначити значення елементам A[i][j] для
// j = i + 1, …, n - 1.
for (int i = 0; i < n; ++i)
	for (int j = i + 1; j < n; ++j)
		A[i][j] = 0;

// Аналогічно присвоюємо значення 2 елементам A[i][j] для
// j = 0, …, i - 1:
for (int i = 0; i < n; ++i)
	for (int j = 0; j < i; ++j)
		A[i][j] = 2;

// Друк матриці
printf("Спосіб 2 (без if):\n");
for (int i = 0; i < n; i++)
{
	for (int j = 0; j < n; j++) printf("%i ",
	A[i][j]);
	printf("\n");
}
printf("\n");

// Можна також зовнішні цикли об'єднати в один
// і отримати ще одне, більш компактне рішення:
int j;
// Заповнюємо рядок з номером i
for (int i = 0; i < n; ++i) {
// Спочатку пишемо 2 нижче діагоналі
	for (j = 0; j < i; ++j)
		A[i][j] = 2;
// Після завершення попереднього циклу i == j, пишемо 1
	A[i][j] = 1;
// Цикл починаємо з збільшення j на 1
	for (++j; j < n; ++j)
// Записуємо 0 вище діагоналі
		A[i][j] = 0;
}
```

```
        // Друк матриці
        printf("Спосіб 3:\n");
        for (int i = 0; i < n; i++)
        {
                for (int j = 0; j < n; j++) printf("%i ",
                A[i][j]);
                printf("\n");
        }
        printf("\n");
}
```

2) Необхідно згенерувати двовимірний масив 5 на 5, що складається з цілих чисел від 0 до 100, в якому потрібно знайти:

а) суму елементів на головній діагоналі (ГД), які > 20;

б) добуток елементів на побічної діагоналі (ПД), які кратні 4;

в) кількість елементів під ПД, які дорівнюють 15;

г) кількість парних елементів над ГД.

Вирішення задачі 2:

```
#include <iostream>
#include <time.h>
#include <iomanip> // для форматування виводу cout
#include "windows.h";
using namespace std;

const int N = 5;

int main() {
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        srand(time(0));
        int matr[N][N];
        while (true)
        {
                // Заповнення матриці і виведення на друк
                cout << "Початкова матриця: " << endl;
                for (int i = 0; i < N; i++)
                {
                        for (int j = 0; j < N; j++)
```

```
                {
                        matr[i][j] = rand() % 99;
                        /*
                        Форматування при використанні cout
                        cout.setf(ios::fixed);
                        cout << setw(2) << matr[i][j]
                        << " ";
                        */
                        printf("%2i ", matr[i][j]);
                }
                cout << endl;
        }
        cout << endl;
        // Введення змінних
        int sGD = 0; // Сума елементів на ГД
        int pPD = 1; // Добуток елементів на ПД
        int cPD = 0; // Кількість елементів під ПД = 15
        int cGD = 0; // Кількість парних під ГД
        for (int i = 0; i < N; i++)
        {
                for (int j = 0; j < N; j++)  {
                        if ((i == j) && (matr[i][j] > 20))
                                sGD += matr[i][j];
                        if (((i + j) == (N - 1)) &&
                                (matr[i][j] % 4 == 0))
                                pPD *= matr[i][j];
                        if (((i + j) > (N - 1)) &&
                                (matr[i][j] == 15)) cPD++;
                        if ((i > j) && (matr[i][j] % 2 ==
                        0))
                                cGD++;
                }
        }
        // Друк результатів
cout << "Сума елементів на ГД > 20 = " << sGD << endl;
cout << "Добуток елементів на ПД кратні 4 = " << pPD
        << endl;
cout << "Кількість елементів під ПД = 15 = " << cPD
        << endl;
cout << "Кількість парних під ГД = " << cGD << endl;
        getchar();
        system("cls");
}
return 0;
```

```
}
```

3) Необхідно знайти мінімальний і максимальний елементи довільної матриці і переставити їх місцями.

Вирішення задачі 3:

```
#include <iostream>
#include <iomanip> // для форматування виводу cout
#include "windows.h";
using namespace std;

const int N = 10;

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	srand(time(0));
	int mass[N][N], max, min;
	// Заповнення матриці і вивід на друк
	cout << "Початкова матриця: " << endl;
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < N; j++)
		{
			mass[i][j] = rand() % 99 - 50;
			cout.setf(ios::fixed);
			cout << setw(3) << mass[i][j] << " ";
		}
		cout << endl;
	}
	cout << endl;
	max = mass[0][0];
	min = mass[0][0];
	int imax, jmax, imin, jmin;
	for (int i = 0; i < N; i++)
		for (int j = 0; j < N; j++)
		{
			if (max < mass[i][j])
			{
				max = mass[i][j];
				imax = i;
				jmax = j;
			}
```

```
                        if (min > mass[i][j])
                        {
                                min = mass[i][j];
                                imin = i;
                                jmin = j;
                        }
                }
        cout << "Min[" << imin << "][" << jmin << "] "
                << min << endl;
        cout << "Max[" << imax << "][" << jmax << "] "
                << max << endl;
        // Заміна місцями
        int t = mass[imax][jmax];
        mass[imax][jmax] = mass[imin][jmin];
        mass[imin][jmin] = t;
        // Друк матриці з заміною
        for (int i = 0; i < N; i++)
        {
                for (int j = 0; j < N; j++)
                {
                        cout.setf(ios::fixed);
                        cout << setw(3) << mass[i][j] << " ";
                }
                cout << endl;
        }
        cout << endl;
        return 0;
}
```

4) З двовимірного масиву необхідно зробити три одновимірних масиву, так, щоб в масив B потрапили елементи, значення яких більше 0, в масив C – менше 0, в масив D – кратні 3.

Вирішення задачі 4:

```
#include <iostream>
#include <time.h>
#include "windows.h";
using namespace std;

const int N = 5; // розмір квадратної матриці

int main()
{
```

```
SetConsoleCP(1251);
SetConsoleOutputCP(1251);
srand(time(0));
int matr[N][N];
int B[N * N], C[N * N], D[N * N];
while (true)
{
	// Заповнення матриці і вивід на друк
	cout << "Початкова матриця: " << endl;
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < N; j++)
		{
			matr[i][j] = rand() % 99 - 50;
			printf("%3i ", matr[i][j]);
		}
		cout << endl;
	}
	cout << endl;
	// Обнулення результуючих масивів
	for (int i = 0; i < N * N; i++)
	{
		B[i] = 0;
		C[i] = 0;
		D[i] = 0;
	}
	// Ініціалізація індексів результуючих масивів
	int iB = 0; // Індекс масиву B
	int iC = 0; // Індекс масиву C
	int iD = 0; // Індекс масиву D
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < N; j++)
		{
			if (matr[i][j] > 0)

				B[iB++] = matr[i][j];
			else if (matr[i][j] < 0)
				C[iC++] = matr[i][j];
			if (matr[i][j] % 3 == 0)
				D[iD++] = matr[i][j];
		}
	}
	// Друк результатів
	printf("B > 0 :\n");
```

```
                for (int i = 0; i < iB; i++)
                        printf("%3i ", B[i]);
                printf("\nC < 0 :\n");
                for (int i = 0; i < iC; i++)
                        printf("%3i ", C[i]);
                printf("\nD - кратні 3 :\n");
                for (int i = 0; i < iD; i++)
                        printf("%3i ", D[i]);
                cout << endl;
                getchar();
                system("cls");
        }
        return 0;
}
```

5) Необхідно порахувати суму елементів в кожному рядку, і добуток елементів в кожному стовпці довільного масиву

Вирішення задачі 5:

```
#include <iostream>
#include <time.h>
#include "windows.h";
using namespace std;

// Оголошуємо розмірність масиву
const int numRows = 3;
const int numCols = 4;

int main() {
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        srand(time(0));
        int matr[numRows][numCols];
        while (true)
        {
                int Sum[numRows];
                int Pr[numCols];
                // Заповнення матриці і вивід на друк
                cout << "Початкова матриця " << numRows
                        << "x" << numCols << ":" << endl;
                for (int i = 0; i < numRows; i++)
                {
                        for (int j = 0; j < numCols; j++)
```

```
                {
                        matr[i][j] = rand() % 10 - 5;
                        printf("%3i ", matr[i][j]);
                }
                cout << endl;
        }
        cout << endl;
        // Ініціалізація масиву сум
        for (int i = 0; i < numRows; i++) Sum[i] = 0;
        // Ініціалізація масиву добутків
        for (int j = 0; j < numCols; j++) Pr[j] = 1;
        // Підрахунок суми
        for (int i = 0; i < numRows; i++)
        {
                for (int j = 0; j < numCols; j++)
                        Sum[i] += matr[i][j];
        }
        // Підрахунок добутків
        for (int j = 0; j < numCols; j++)
        {
                for (int i = 0; i < numRows; i++)
                        Pr[j] *= matr[i][j];
        }
        // Друк результатів
        printf("Масив сум елементів ");
        printf("в кожному рядку:\n");
        for (int i = 0; i < numRows; i++)
                printf("%3i\n", Sum[i]);
        printf("Масив добутків елементів ");
        printf("в кожному стовпці:\n");
        for (int j = 0; j < numCols; j++)
                printf("%3i ", Pr[j]);
        printf("\n");
        getchar();
        system("cls");
    }
    return 0;
}
```

6) Необхідно знайти максимальний добуток елементів в кожному стовпці.

Вирішення задачі 6:

```
#include <iostream>
```

```
#include <time.h>
#include "windows.h";
using namespace std;

// Оголошуємо розмірність масиву
const int numRows = 3;
const int numCols = 4;

int main() {
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	srand(time(0));
	int matr[numRows][numCols];
	while (true)
	{
		int Sum[numRows];
		int Pr[numCols];
		// Заповнення матриці і вивід на друк
		cout << "Початкова матриця " << numRows
			<< "x" << numCols << ":" << endl;
		for (int i = 0; i < numRows; i++)
		{
			for (int j = 0; j < numCols; j++)
			{
				matr[i][j] = rand() % 10 - 5;
				printf("%3i ", matr[i][j]);
			}
			cout << endl;
		}
		cout << endl;
		// Ініціалізація масиву добутків
		for (int j = 0; j < numCols; j++) Pr[j] = 1;
		// Підрахунок добутків
		int max = INT_MIN;
		for (int j = 0; j < numCols; j++)
		{
			for (int i = 0; i < numRows; i++)
				Pr[j] *= matr[i][j];
			if (Pr[j] > max) max = Pr[j];
		}
		// Друк результатів
		printf("Масив добутків елементів ");
		printf("в кожному стовпці:\n");
		for (int j = 0; j < numCols; j++)
			printf("%3i ", Pr[j]);
```

```
            printf("\n");
            printf("Максимальне значення добутку : %i\n", max);
            getchar(); system("cls");
        }
        return 0;
}
```

7) Написати програму, яка вводить по рядках з клавіатури двовимірний масив і обчислює суму його елементів по стовпчиках.

Вирішення задачі 7:

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "windows.h";
#include <conio.h>

#define ROW 3 // кількість рядків
#define COL 5 // кількість стовпців

void main()
{
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        int a[ROW][COL]; // масив
        int s[COL];      // сума елементів
        int i, j;
        printf("\nВведіть масив\n");
        printf("Після введення елементів кожного рядка,");
        printf("\n%i цілих чисел, натискайте <Enter>\n", COL);
        for (i = 0; i < ROW; i++) // ROW рядків
        {
                printf("->");
                for (j = 0; j < COL; j++)
                        scanf("%i", &a[i][j]);
        }
        printf("\nВведений масив\n");
        for (i = 0; i < ROW; i++)
        {
                for (j = 0; j < COL; j++)
                        printf("%i ", a[i][j]);
                printf("\n");
        }
        // "очистимо" масив s
```

```
                for (i = 0; i < COL; i++)
                        s[i] = 0;
        // обробка
        for (j = 0; j < COL; j++) // для кожного стовбцю
                for (i = 0; i < ROW; i++) // підсумуємо елементи
                        s[j] += a[i][j];
        printf("----------------------------\n");
        for (i = 0; i < COL; i++)
                printf("%i ", s[i]);
        printf("\nДля завершення натисніть <Enter>");
        _getch();
}
```

8) Написати програму, яка вводить по рядках з клавіатури двовимірний масив і обчислює суму його елементів по рядках.

Зробити самостійно.

9) Написати програму, яка обробляє результати іспиту. Для кожної оцінки програма повинна обчислити відсоток від загальної кількості оцінок. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним).

```
Обробка результатів іспиту
Введіть вихідні дані:
п'ятірок -> 12
четвірок -> 10
трійок -> 7
двійок -> 1
Результати іспиту:
п'ятірок 12 40%
четвірок 10 33%
трійок    7 23%
двійок    1  3%
Для завершення натисніть <Enter>
```

Вирішення задачі 9:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "windows.h"
```

```
void main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int n[6];   // кількість двійок, ... п'ятірок
	int s = 0;  // всього оцінок
	float p[6]; // відсоток кожної оцінки
	char mes[6][10];
	strcpy(mes[0], "\0");
	strcpy(mes[1], "\0");
	strcpy(mes[2], "двійок\0");
	strcpy(mes[3], "трійок\0");
	strcpy(mes[4], "четвірок\0");
	strcpy(mes[5], "п'ятірок\0");
	int i;
	puts("Обробка результатів іспиту");
	puts("Введіть початкові дані:");
	for (i = 5; i >= 2; i--)
	{
		printf("%s ->", mes[i]);
		scanf("%i", &n[i]);
		s += n[i];
	}
	// обчислимо відсоток кожної оцінки
	for (i = 2; i < 6; i++)
		p[i] = (float)n[i] / s * 100;
	puts("Результати іспиту");
	puts("------------------------------");
	for (i = 5; i >= 2; i--)
		printf("%8s %2i   %2.0f%% \n", mes[i], n[i],
p[i]);
	puts("------------------------------");
	puts("Для завершення натисніть <Enter>");
	getchar();
}
```

10) Написати програму, яка перевіряє, чи є введена з клавіатури квадратна матриця “магічним квадратом”. Магічним квадратом називається матриця, у якої сума чисел у кожному горизонтальному ряду, в кожному вертикальному і по кожній з діагоналей одна і та ж (як показано на рис. 7.1).

2	9	4
7	5	3
6	1	8

13	8	12	1
2	11	7	14
3	10	6	15
16	5	9	4

Рисунок 7.1

Вирішення задачі 10:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "windows.h"

#define SZ 5 // максимальний розмір матриці

void main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int a[SZ][SZ]; // матриця
	int n;                // розмір матриці що перевіряється
	int ok;               // матриця - "магічний" квадрат"
	int i, j;             // індекси масиву
	int sum;              // сума елементів головної діагоналі
	int temp;             // сума елементів поточного рядка,
                          // стовпчика
                          // або другий діагоналі матриці
	printf("*** МАГІЧНИЙ КВАДРАТ ***\n");
	printf("\nВведіть розмір матриці (3..%i) -> ", SZ);
	scanf("%i", &n);
	printf("Введіть рядки матриці\n");
	printf("Після введення рядка %i цілих чисел, ", n);
	printf("натискайте <Enter>\n");
	for (i = 0; i < n; i++)
	{
		printf("->");
		for (j = 0; j < n; j++)
			scanf("%i", &a[i][j]);
	}
	ok = 1; // нехай матриця - "магічний" квадрат
	// обчислимо суму елементів головної діагоналі
	sum = 0;
	for (i = 0; i < n; i++)
```

```
            sum += a[i][i];
        // обчислюємо суми по рядках
        i = 0;
        do {
                temp = 0; // сума елементів поточного рядка
                for (j = 0; j < n; j++)
                        temp += a[i][j];
                if (temp != sum) ok = 0;
                i++;
        } while (ok && i < n);

        if (ok)         {
                // тут сума елементів кожного рядка
                // дорівнює сумі елементів головної діагоналі
                // обчислюємо суми по стовпцях
                j = 0;
                do {
                        temp = 0; // сума ел-тів поточного стовпця
                        for (i = 0; i < n; i++)
                                temp += a[i][j];
                                if (temp != sum) ok = 0;
                        j++;
                } while (ok && i < n);
        }
        if (ok) {
                // тут сума елементів кожного рядка
                // дорівнює сумі елементів кожного стовпця
                // і сумі елементів головної діагоналі.
                // обчислимо суму елементів другої
                // головної діагоналі
                temp = 0;
                i = n - 1;
                for (j = 0; j < n; j++)
                        temp += a[i--][j];
                if (temp != sum) ok = 0;
        }
        printf("введена матриця ");
        if (!ok)
                printf("не ");
        printf("є магічним квадратом.\n");
        printf("\nДля завершення натисніть <Enter>");
        getchar();
}
```

11)Написати програму, яка обчислює дохід за вкладом. Процентна ставка за вкладом визначається на основі даних, наведених у таблиці 7.1.

Таблиця 7.1 – Початкові дані до розрахунку доходу за вкладом.

Сума, тис. грн.	Строк владу та процентна ставка					
	3 міс.	6 міс.	12 міс.	18 міс.	24 міс.	36 міс.
до 60	15.0%	16.5%	18.0%	19.5%	21.0%	22.0%
до 250	16.5%	18.0%	19.5%	21.0%	22.5%	24.0%
понад 250	18.5%	19.5%	21.0%	22.5%	24.0%	27.0%

Вирішення задачі 11:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "windows.h"

void main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	// процентна ставка (таблиця 7.1)
	float rate[3][6] =
	{
	15.0,16.5,18.0,19.5,21.0,24.0,
	16.5,18.0,19.5,21.0,22.5,24.0,
	18.0,19.5,21.0,22.5,24.0,27.0
	};
	float value;  // сума
	int period;           // строк (місяців)
	float profit; // дохід
	int r, c;
	// вивести таблицю
	for (int r = 0; r < 3; r++)
	{
		for (int c = 0; c < 6; c++)
			printf("%8.2f", rate[r][c]);
		printf("\n");
	}
	printf("Сума, грн.-> ");
	scanf("%f", &value);
	printf("Строк, міс. -> ");
	scanf("%i", &period);
	if (value <= 10000)
```

```
            r = 0;
        else
            if (value <= 300000)
                r = 1;
            else
                r = 3;
        switch (period)
        {
        case 3: c = 0; break;
        case 6: c = 1; break;
        case 12: c = 2; break;
        case 18: c = 3; break;
        case 24: c = 4; break;
        case 36: c = 6; break;
        default: period = 0;
        }
        if (period != 0)
        {
            printf("\nпроцентна ставка: %5.2f",
                rate[r][c]);
            profit = value * rate[r][c] / 100 / 12 * period;
            printf("\nДохід: %6.2f грн.", profit);
        }
        else
            printf("Неправильно зазначено термін");
        printf("\nДля завершення натисніть <Enter>");
        getchar();
}
```

12) Написати програму, яка обробляє результати спортивних змагань: вишиковує команди відповідно до кількості набраних очок, яка обчислюється за формулою $K = 7N_G + 6N_S + 5N_B$, де N_G, N_S і N_B – кількість золотих, срібних і бронзових медалей. Нижче наведено рекомендований вид екрану програми (дані, введені користувачем, виділені напівжирним).

```
Введіть в одному рядку кількість золотих, срібних і бронзових медалей
Австрія-> 0 1 1
Німеччина-> 1 0 2
Норвегія-> 4 2 1
Україна-> 2 3 2
Фінляндія-> 1 2 2
```

```
-----------------------------------
  Команда        Зол.   Сер.   Бр.    Всього Очок
1 Норвегія       4      2      1      7      45
2 Україна        2      3      2      7      42
3 Фінляндія      1      2      2      5      29
4 Німеччина      1      0      2      3      17
5 Австрія        0      1      1      2      11
```

Вирішення задачі 12:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "windows.h"

#define NC 5 // число команд

void main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	char team[5][11];
	strcpy(team[0], "Австрия\0");
	strcpy(team[1], "Германия\0");
	strcpy(team[2], "Норвегия\0");
	strcpy(team[3], "Україна\0");
	strcpy(team[4], "Финляндия\0");
	// таблиця результатів
	int result[NC + 1][6];
	// result[i][0] - золотих
	// result[i][1] - срібних
	// result[i][2] - бронзових
	// result[i][3] - всього
	// result[i][4] - очок
	// result[i][5] - номер команди
	// NC + 1-й рядок використовується як буфер
	// при сортуванні таблиця
	int i, j;
	int max; // номер рядка таблиці, в якій
	// кількість очок максимально
	printf("Введіть в одному рядку");
	printf("кількість золотих,\n");
```

```
printf("серебряних і бронзових медалей\n");
// введення початкових даних
for (i = 0; i < NC; i++) {
	printf("%s ->", team[i]);
	scanf("%i%i%i", &result[i][0], // золотих
		&result[i][1],// срібних
		&result[i][2]); // бронзових
	result[i][5] = i;	// номер команди
}
// обчислимо загальну кількість медалей і очок
for (i = 0; i < NC; i++)
{
	result[i][3] =
		result[i][0] + result[i][1] +
		result[i][2];
	result[i][4] =
		result[i][0] * 7 + result[i][1] * 6 +
		result[i][2] * 5;
}
// сортування рядків (методом простого вибору)
// відповідно до кількості очок
for (i = 0; i < NC + 1; i++)
{
	// в частині таблиці, починаючи з рядка i,
	// знайти j-й рядок, в якій елемент
	// result[j][5] максимальний
	max = i; // нехай це рядок з номером i
	for (j = i + 1; j < NC; j++)
	{
		if (result[j][4] > result[max][4])
			max = j;
	}
	// поміняємо місцями i-й рядок со рядком
	// з номером max
	// в якості буфера використовуємо останній
	// рядок таблиці
	for (j = 0; j < 6; j++)
		result[NC][j] = result[i][j];
	for (j = 0; j < 6; j++)
		result[i][j] = result[max][j];
	for (j = 0; j < 6; j++)
		result[max][j] = result[NC][j];
}
// тут таблиця впорядкована
printf("%3s%12s%8s%8s%8s%8s%8s",
```

```
                "", "Команда",
                "Золото", "Срібло", "Бронза",
                "Всього", "Очок");
        for (i = 0; i < NC; i++) {
                printf("\n%12s", team[result[i][5]]);
                for (j = 0; j < 5; j++)
                        printf("%8i", result[i][j]);
        }
        printf("\nДля завершення натисніть <Enter>");
        getchar();
}
```

Практична робота 8

ФУНКЦІЇ. ПЕРЕДАЧА МАСИВІВ У ФУНКЦІЮ

Мета: Оволодіння практичними навичками роботи з функціями в мовах C/C++

Теми для попереднього опрацьовування

– Функції в мовах C/C++
– Робота з масивами

Теоретичні відомості

Функції

Функція – це послідовність операцій для виконання певного завдання. Часто програми переривають виконання одних функцій заради виконання інших. Програми на мові C ++ працюють схожим чином. Іноді, коли програма виконує код, вона може зіткнутися з викликом функції. Виклик функції – це вираз, який вказує процесору перервати виконання поточної функції і приступити до виконання іншої функції. Процесор "залишає закладку" в поточній точці виконання, а потім виконує функцію, що викликається. Коли виконання функції, що викликається завершено, процесор повертається до закладки і відновлює виконання перерваної функції.

Функція, в якій знаходиться виклик, називається *caller*, а функція, яку викликають – функцією, що викликається.

Параметри і аргументи функцій

У багатьох випадках нам потрібно буде передавати дані в функцію, що викликається, щоб вона могла з ними якось взаємодіяти. Наприклад, якщо ми хочемо написати функцію множення двох чисел, то нам потрібно якимось чином повідомити функції, які це будуть числа. В іншому випадку, як вона дізнається, що на що перемножать? Тут нам на допомогу приходять параметри і аргументи.

Параметр функції – це змінна, яка використовується в функції, і значення якої надає функція, яка викликає (*caller*). Параметри вказуються при оголошенні функції в круглих дужках. Якщо їх багато, то вони перераховуються через кому. Параметри кожної функції дійсні тільки всередині цієї

функції.

Аргумент функції – це значення, яке передається з функції, що викликає, в функцію яку викликають, і яке вказується в дужках при виклику функції.

При виконанні функції, всі її параметри створюються як локальні змінні, а значення кожного з аргументів копіюється в відповідний параметр (локальну змінну). Цей процес називається передачею за значенням. Таким чином будь-які зміни з аргументом, зроблені всередині функції, не відображаються на ньому.

Однак механізм передачі масивів у функцію інший. Це буде розглянуто в подальшому, а на даному етапі необхідно пам'ятати, що на відміну від передачі за значенням, будь-які зміни з масивом, зроблені всередині функції, відобразяться на масиві, який переданий в функцію як аргумент.

Задачі

Загальні умови:

Кожне завдання містить приклад роботи з функціями мов C/C++, і може містити умови вирішення. Після списку завдань надано рішення у вигляді програмного коду.

1) Написати функцію, яка обчислює об`єм циліндра.

Вирішення задачі 1:

```
#include <iostream>
#include "windows.h";
using namespace std;

const double pi = 3.14;

double volume(int r, int h)
{
	double v = pi * pow(r, 2) * h;
	return v;
}

int main()
{
```

```
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	while (true)
	{
		// Друк результатів
		int r, h;
		cout << "Введіть r: ";
		cin >> r;
		cout << "Введіть h: ";
		cin >> h;
		// Друк результату
		cout << "Об'єм дорівнює = "
			<< volume(r, h) << endl;

		system("pause");
		system("cls");
	}
	return 0;
}
```

2) Написати функцію, яка порівнює два числа і виводить знак порівняння.

Вирішення задачі 2:

```
#include <iostream>
#include "windows.h";
using namespace std;

char Sr(int a, int b)
{
	char rez;
	if (a > b) rez = '>';
	else if (a < b) rez = '<';
	else rez = '=';
	return rez;
}

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
```

```
	while (true)
	{
		// Введення початкових даних
		int a, b;
		cout << "Введіть a: ";
		cin >> a;
		cout << "Введіть b: ";
		cin >> b;
		// Друк результату
		cout << "a " << Sr(a, b) << " b" << endl;

		system("pause");
		system("cls");
	}
	return 0;
}
```

3) Написати функцію, яка обчислює опір ланцюга. Параметри функції: два опору, тип ланцюга (паралельне з'єднання або послідовне)

Вирішення задачі 3:

```
#include <iostream>
#include "windows.h";
using namespace std;

float R(int R1, int R2, bool type)
{
	int r;
	switch (type)
	{
	// для паралельного ланцюга
	case 0: r = R1 * R2 / (R1 + R2);
		break;
	// для послідовного ланцюга
	case 1: r = R1 + R2;
		break;
	default:
		break;
	}
	return r;
}

int main()
```

```
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	while (true)
	{
		// Введення початкових даних
		int R1, R2, type;
		cout << "Введіть R1: ";
		cin >> R1;
		cout << "Введіть R2: ";
		cin >> R2;
		cout << "Введіть тип ланцюга (0 - паралельний, ";
		cout << "1 - послідовний):";
		cin >> type;
		// Друк результату
		cout << "Опір ланцюга = " << R(R1, R2, type)
			<< endl;

		system("pause");
		system("cls");
	}
	return 0;
}
```

4) Написати функцію, яка виводить рядок з заданого символу заданої довжини

Вирішення задачі 4:

```
#include <iostream>
#include "windows.h";
using namespace std;

void print_string(int lng, char c)
{
	for (int i = 0; i < lng; i++)
	{
		cout << c;
	}
	cout << endl;
}

int main()
```

```
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	while (true)
	{
		// Введення початкових даних
		int lng;
		char ch;
		cout << "Введіть символ: ";
		cin >> ch;
		cout << "Введіть довжину рядка: ";
		cin >> lng;
		// Друк результату
		cout << "Рядок : " << endl;
		print_string(lng, ch);

		system("pause");
		system("cls");
	}
	return 0;
}
```

5) Написати функцію, яка знаходить факторіали чисел від 0 до 8.

Вирішення задачі 5:

```
#include <iostream>
#include "windows.h";
using namespace std;

unsigned int fact(int x)
{
	unsigned int f = 1;
	for (int i = 2; i <= x; i++)
	{
		f *= i;
	}
	return f;
}

int main()
{
	SetConsoleCP(1251);
```

```
	SetConsoleOutputCP(1251);

	while (true)
	{
		// Введення початкових даних
		unsigned int x;
		cout << "Введіть число: ";
		cin >> x;
		// Друк результату
		cout << "Факторіал " << x << " = "
			<< fact(x) << endl;
		// Друк факторіалів чисел від 0 до 8
		cout << "Факторіали чисел від 0 до 8 " << endl;
		for (int i = 0; i < 9; i++)
		{
			cout << i << "! = " << fact(i) << endl;
		}

		system("pause");
		system("cls");
	}
	return 0;
}
```

6) Написати функцію, яка досліджує можливість вирішення квадратного рівняння і видає:

2 – якщо коренів 2;

1 – якщо корінь 1;

0 – якщо дискримінант меньше 0;

–1 – якщо помилка вихідних даних (a = 0).

Вирішення задачі 6:

```
#include <iostream>
#include "windows.h";
using namespace std;

int KU(float a, float b, float c)
{
	float d = 1.;
	if (a == 0) return -1;		// не вирішуємо, так як
					// не знаходимо x1 і x2
	d = b * b - 4 * a * c;
```

```
        if (d < 0) return 0;
        else if (d == 0) return 1;   // т.е. x1 = x2
        else return 2;
}

int main()
{
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);

        float a, b, c;
        while (true)
        {
                // Введення початкових даних
                cout << "Введіть a, b, c :" << endl;
                cin >> a >> b >> c;

                // Друк результату
                switch (KU(a, b, c))
                {
                case -1: cout << "Помилка вихідних даних";
                        break;
                case  0: cout << "D < 0, немає коренів"; break;
                case  1: cout << "D = 0, x1 = x2"; break;
                case  2: cout << "2 кореня"; break;
                }
                cout << endl;

                system("pause");
                system("cls");
        }
        return 0;
}
```

7) Написати функцію, яка заповнює одновимірний масив випадковими числами.

Вирішення задачі 7:

```
#include <iostream>
#include "windows.h";
#include <time.h>
using namespace std;
```

```
// Прототип
void Zap(int massiv[], int n);

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	srand(time(0));
	float a, b, c;
	while (true)
	{
		const int n = 10;
		int mas[n];

		// Заповнення масиву
		Zap(mas, n);

		// Друк результату
		for (int i = 0; i < n; i++)
		{
			cout << mas[i] << " ";
		}
		cout << endl;

		system("pause");
		system("cls");
	}
	return 0;
}

void Zap(int massiv[], int n)
{
	for (int i = 0; i < n; i++)
	{
		massiv[i] = rand() % 10;
	}
}
```

8) Написати функцію, яка знаходить максимум в масиві.

Вирішення задачі 8:

```
#include <iostream>
#include "windows.h";
```

```
#include <time.h>
using namespace std;

// Прототипи
void Zap(int massiv[], int n);
int Max(int massiv[], int n);

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	srand(time(0));

	float a, b, c;

	while (true)
	{
		const int n = 10;
		int mas[n];
		// Заповнення масиву
		Zap(mas, n);
		// Друк результату
		for (int i = 0; i < n; i++)
		{
			cout << mas[i] << " ";
		}
		cout << endl;
		cout << "Максимальний елемент масиву = "
			 << Max(mas, n) << endl;

		system("pause");
		system("cls");
	}
	return 0;
}

void Zap(int massiv[], int n)
{
	for (int i = 0; i < n; i++)
	{
		massiv[i] = rand() % 10;
	}
}

int Max(int massiv[], int n)
```

```
{
	int m = INT_MIN;
	for (int i = 0; i < n; i++)
	{
		if (massiv[i] > m) m = massiv[i];
	}
	return m;
}
```

9) Виконати наступні дії:

– передати одновимірний масив у функцію;
– у функції зробити з нього двовимірний;
– провести з ним перетворення (віддзеркалити);
– вивести на друк новий двовимірний масив;
– перетворити його в одновимірний;
– вивести (повернути) одновимірний масив з функції.

Вирішення задачі 9:

```
#include <iostream>
#include "windows.h";
using namespace std;

#define N 3
#define M 3

// Прототип
void Fun(int mas[]);

int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);

	int mas1[N * M] = { 4, 9, 8, 6, 4, 8, 5, 3, 2 };

	// Друк вихідного одновимірного масиву
	cout << "Вихідний одновимірний масив: " << endl;
	for (int i = 0; i < N * M; i++)
		cout << mas1[i] << " ";
	cout << endl;
```

```
	// Перетворення масиву в функції
	Fun(mas1);

	// Друк одновимірного масиву після функції
	cout << "Одновимірний масив після функції: ";
	cout << endl;
	for (int i = 0; i < N * M; i++)
		cout << mas1[i] << " ";
	cout << endl;

	system("pause");
	return 0;
}

void Fun(int mas[])
{
	int mas2[N][M];
	cout << "Двовимірний з одновимірного:" << endl;
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < M; j++)
		{
			mas2[i][j] = mas[i * M + j];
			cout << mas2[i][j] << " ";
		}
		cout << endl;
	}

	// Дзеркальне відображення
	int t;
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < M / 2; j++)
		{
			t = mas2[i][j];
			mas2[i][j] = mas2[i][M - 1 - j];
			mas2[i][M - 1 - j] = t;
		}
	}

	// Друк нового двовимірного масиву
	cout << "Двовимірний масив після ";
	cout << "дзеркального відображення : " << endl;
	for (int i = 0; i < N; i++)
	{
```

```
                for (int j = 0; j < M; j++)
                {
                        cout << mas2[i][j] << " ";
                }
                cout << endl;
        }

        // Перетворення двовимірного в одновимірний
        for (int i = 0; i < N; i++)
        {
                for (int j = 0; j < M; j++)
                {
                        mas[i * M + j] = mas2[i][j];
                }
        }
} // end Fun
```

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мова програмування С. Лабораторний практикум: Навч.-методичний посібник / Дерев'янко О.С., Любченко Н.Ю., Клименко А.М., Сомхієва О.С. – Харків : НТУ "ХПІ", 2011. – 196 с.
2. Культин Н. Б. C/C++ в задачах и примерах. – 3-е изд., доп. и исправл. – СПб.: БХВ-Петербург, 2019. –272 с.
3. Режим доступу : https://ravesli.com/ – Уроки программирования на языке С++. – Дата звернення: 20.11.2020.
4. Страуструп Б. Дизайн и эволюция языка С++ / Б. Страуструп; пер. с англ. Слинкина А. А. – М.: ДМК Пресс, 2016. - 446 с.
5. Керниган Б. В. Язык программирования Си / Б. В. Керниган, Д. М. Ритчи; пер. с англ. С. Штаркмана – 3-е изд. – СПб.: Невский Диалект, 2001.– 352 с.

ЗМІСТ

Навчальне видання

СОБОЛЬ Максим Олегович
ЛЮБЧЕНКО Наталія Юріївна
ПАРЖИН Юрій Володимирович
ПУГАЧОВ Роман Володимирович

ОСНОВИ ПРОГРАМУВАННЯ НА С/С++ В ПРИКЛАДАХ

Частина 1

Навчально-методичний посібник
для студентів комп'ютерних спеціальностей
вищих навчальних закладів

Роботу рекомендував до видання проф. Гамаюн І.П.
Відповідальний за випуск доц. Галкин С.О.

Дизайн обкладинки Пугачов Р.В.

В авторській редакції

Підписано до друку 26.02.21. Формат 60x84 1/16. Папір офсет.
Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 2,8
Наклад 300 прим. Зам №. Ціна договірна

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