ЛАБОРАТОРНА РОБОТА № 1

Если ваша работа не документирована, значит вы не работали.
Ада Лавлейс

ВИВЧЕННЯ МОЖЛИВОСТЕЙ СЕРЕДОВИЩА MICROSOFT VISUAL C++ зі СТВОРЕННЯ, ТРАНСЛЯЦІЇ ТА НАЛАГОДЖЕННЯ ПРОГРАМ

1.1 Мета роботи

Вивчення можливостей середовища Microsoft Visual C++. Створення, трансляція та налагодження програм. Отримання практичних навичок алгоритмізації задач та створення програм з використанням мови C++.

1.2 Методичні матеріали до виконання лабораторної роботи

Пді час підготовки до лабораторної работи повторити матеріал наступних лекцій:

Лекція 3. Основи програмування мовою C++

Лекція 4. Керуючі конструкції мови C++

Контрольні питання:

1. Типи данних у C++.
2. Оголошення змінних та констант в C++. Особливості ініціалізації змінних.
3. Правила узгодження типів у C++
4. Арифметичні, логічні та бітові операції в C++.
5. Пріоритет та асоціативність операцій в C++.
6. Керування ходом виконання програми в C++.
7. Умовний оператор. Повна та неповна форма.
8. Оператор вибору альтернатив.
9. Види циклів. Цикл з передумовою, с постумовою та з параметром.
10. Оператори переходів. Оператори виходу з циклу.
11. Принципи структурного програмування.
12. Особливості налагодження програм в середовищі Microsoft Visual C++.

1.3 Організація самостійної роботи студентів

Під час підготовки до виконання лабораторної роботи необхідно вивчити індивідуальне завдання (п. 1.4), виконати розробку алгоритму вирішення задачі та підготовити текст програми щодо реалізації розробленого алгоритму, підготовити відповідні розділи звіту та вивчити відповідний теоретичний матеріал, який викладено у лекціях, підручниках і навчальних посібниках

При вивченні теоретичного матеріалу необхідно усвідомити, що будь-яка програма (проект) на C++ складається з файлів. Файли транслюються C-компілятором незалежно один від одного, а потім поєднуються програмою-компоновником задач. За наслідками цього створюється файл із

програмою, готовою до виконання. У мові C++ вихідні файли бувають двох типів:

- заголовні, або h-файли;
- файли реалізації, або C-файли.

Назви заголовних файлів мають розширення ".h". Назви файлів реалізації мають розширення ".c" для мови C й ".cpp", ".cxx" або ".cc" для мови C++. Заголовні файли містять тільки описи. Файли реалізації, або C-файли, містять тексти функцій і визначення глобальних змінних. Говорячи спрощено, C-файли містять самі програми, а h-файли - лише інформацію про програми.

Подання вихідних текстів у вигляді заголовних файлів і файлів реалізації необхідно для створення великих проектів, що мають модульну структуру. Заголовні файли використовуються для передачі інформації між модулями. Файли реалізації – це окремі модулі, які розробляються та транслюються незалежно одне від одного та поєднуються при створенні виконуваної програми. Файли реалізації підключають заголовні файли за допомогою директиви препроцесора *#include*. Самі заголовні файли також можуть використовувати інші заголовні файли.

Функція є основною структурною одиницею мови C. В інших мовах функції називають підпрограмами. Функція – це фрагмент програми, що може викликатися з інших програм. Функція – зазвичай, алгоритм, що описується та реалізується окремо від інших алгоритмів. При виклику функції передаються аргументи, які можуть бути використані в тілі функції. За результатами роботи функція повертає значення деякого типу.

Розглянемо приклад програми, яка виводить на екран фразу *"Hello, World"*.

```
#include <iostream>
int main()    {
    cout<<"Hello, World\n";
    return 0; }
```

Перший рядок підключає заголовний файл із описом стандартних функцій введення даних. У файлі описаний прототип функції `cout`. Виконання програми починається з функції *`main`,* яка повертає після закінченні роботи ціле число (рядок *`return 0;`*), що трактується операційною системою, як код завершення завдання. Число нуль означає успішне виконання задачі, але програміст може самостійно визначати коди завершення.

При розгляді типів змінних у C++ варто розрізняти поняття базового типу та конструкції, що дозволяє будувати нові типи на основі вже побудованих. Базових типів зовсім небагато – це цілі та дійсні числа, які розрізняються за діапазонами можливих значень (або по довжині в байтах) та логічний тип даних. До конструкцій відноситься масив, вказівник та структура, а також клас.

Цілочисельні типи розрізняються по довжині в байтах і по наявності знака. Їх чотири – *`char, short, int і long`*. Крім того, до опису можна додавати модифікатори *`unsigned`* або *`signed`* для беззнакових (додатних) або знакових цілих чисел.

Дійсних типів даних два: довгі дійсні числа *`double`* (тобто "подвійна точність") та коротке дійсне число *`float`* (тобто, з рухомою крапкою). Дійсне

число типу *double* займає 8 байтів, типу *float* – 4 байти. Тип *double* є основним для комп'ютера.

У мові C++ уведено логічний тип *bool*. Змінні типу *bool* приймають два значення: *false* та *true*, які є ключовими словами мови.

При вивченні типів даних необхідно звернути увагу на те, як вони представлені у машинному форматі та особливості їх опису у програмах.

Для створення нових типів у C++ використовуються конструкції масиву, вказівника та структури.

Опис масиву складається з назви базового типу, назви масиву та його розміру, що вказується у квадратних дужках. Розмір масиву обов'язково повинен бути цілочисельною константою або константним виразом:

```
int a[10];
char c[256];
double d[1000];
```

У першому рядку описаний масив цілих чисел з 10 елементів. Необхідно пам’ятати, що нумерація в С завжди починається з нуля, тому індекси елементів масиву змінюються в межах від 0 до 9. У другому рядку описаний масив символів з 256 елементів (індекси в межах 0...255), у третьому – масив дійсних чисел з 1000 елементів (індекси в межах 0...999). Для доступу до елемента масиву вказується ім'я масиву та індекс елемента у квадратних дужках, наприклад: `a[10], c[255], d[123]`.

Вказівник – це змінні, які зберігають адреси об'єктів. При описі вказівника треба задати тип об'єктів, адреси яких будуть знаходитися в ньому. Перед ім'ям вказівника при описі ставиться зірочка, щоб відрізнити його від звичайної змінної. Приклади описів вказівників:

```
int *a, *b, c, d;
char *e; void *f;
```

У першому рядку описані вказівники *a* і *b* на тип *int* та прості змінні *c* та *d* типу *int* (*c* і *d* – не є вказівниками!). З вказівниками можливі такі дві дії:

1. Присвоїти вказівнику адресу деякої змінної. Для цього використовується операція узяття адреси, що позначається амперсендом **&**. Наприклад, рядок `a = &c;` дозволяє вказівнику `a` присвоїти значення адреси змінної `c`;
2. Одержати об'єкт, адреса якого записана у вказівнику; для цього використовується операція зірочка «*», що записується перед вказівником. Наприклад, рядок `d = *a;` присвоює змінній `d` значення цілочисельної змінної, адреса якої записана в `a`.

Конструкції масиву та вказівника при описі типу можна застосовувати багаторазово в довільному порядку. Крім того, можна описувати прототип функції. У такий спосіб можна будувати складні конструкції типу "масив вказівників", "вказівник на вказівник", "вказівник на масив", "функція, що повертає значення типу вказівник", "вказівник на функцію" тощо. Правила такі:

- для групування можна використовувати круглі дужки, наприклад, опис `int *(x[10]);` означає "масив з 10 елементів типу вказівник на `int`";

- при відсутності дужок пріоритети конструкцій опису розподілені в такий спосіб:
  - операція * визначення вказівника має найнижчий пріоритет. Наприклад, опис `int *x[10];` означає "масив з 10 елементів типу вказівник на *int*". Тут до імені змінної `x` спочатку застосовується операція визначення масиву `[]` (квадратні дужки), оскільки вона має більше високий пріоритет, чим зірочка. Потім до отриманого масиву застосовується операція визначення вказівника. У результаті виходить "масив вказівників", а не вказівник на масив! Якщо нам потрібно визначити вказівник на масив, то варто використовувати круглі дужки при описі: `int (*x)[10];` Тут до імені `x` спочатку застосовується операція * визначення вказівника;
  - операції визначення масиву `[]` (квадратні дужки після імені) та визначення функції (круглі дужки після ім'я) мають однаковий пріоритет, більше високий, ніж зірочка. Приклади:
    - `int f();` – описано прототип функції `f` без аргументів, що повертає значення типу `int;`
    - `int (*f())[10];` – описано прототип функції `f` без аргументів, що повертає значення типу вказівник на масив з 10 елементів типу `int`;

Останній приклад не є очевидним. Загальний алгоритм розуміння складного опису можна охарактеризувати як **«читання зсередини»**. Спочатку знаходимо описуване ім'я. Потім визначаємо, яка операція застосовується до імені першою. Якщо немає круглих дужок для групування, то це або визначення вказівника (зірочка ліворуч від імені), або визначення масиву (квадратні дужки праворуч від імені), або визначення функції (круглі дужки праворуч від імені). Так виконується перший крок аналізу складного опису. Потім знаходимо наступну операцію опису, що застосовується до вже виділеної частини складного опису, та повторюємо це доти, поки не вичерпаємо весь опис. Розглянемо цей алгоритм на наступному прикладі:

```
void (*a[100])(int x);
```

Описується змінна `a`. До неї спочатку застосовується операція опису масиву з 100 елементів, далі – визначення вказівника, далі – функція від одного цілочисельного аргументу `x` типу `int`, нарешті – визначення типу, що повертається, – `void`. Опис читається в такий спосіб:

1. `a` - це
2. Масив з 100 елементів типу
3. Вказівник на
4. Функцію з одним аргументом `x` типу `int`, що повертає значення типу
5. `void`.

Нижче розставлені номери операцій у порядку їхнього застосування в описі змінної a:

```
void (*  a  [100])(int x);
5)    3) 1) 2)    4)
```

Спеціального типу даних «рядок» у С немає. Рядки представляються масивами символів (а символи – їхніми числовими кодами). Останнім символом

масиву, що представляє рядок, повинен бути символ з нульовим кодом. Приклад:

```
char str[10];
str[0] = 'e'; str[1] = '2';
str[2] = 'e'; str[3] = '4';
str[4] = 0;
```

Описано масив `str` з 10 символів, що може представляти рядок довжиною не більше 9, оскільки один елемент повинен бути зарезервований для завершального нуля. Далі в масив `str` записується рядок `"e2e4"`. Рядок завершується нульовим символом. Усього запис рядка використовує 5 перших елементів масиву `str` з індексами `0...4`. Останні 5 елементів масиву не використовуються. Масив можна ініціалізувати безпосередньо при описі, наприклад:

```
char st[] = "abc";
```

Тут не вказується у квадратних дужках розмір масиву `st`, компілятор його обчислює сам. Після операції присвоювання записана строкова константа `"abc"`, що заноситься в масив `st`. У результаті компілятор створює масив `st` із чотирьох елементів, оскільки на рядок приділяється 4 байти, включаючи завершальний нуль. Строкові константи записуються у подвійних апострофах, на відміну від символьних, які записуються в одинарні.

Вирази у С складаються зі змінних або констант, до яких застосовуються різні операції. Для указання порядку операцій можна використовувати круглі дужки.

Крім звичайних операцій, таких, як додавання або множення, у С існує ряд операцій, незвичних для початківців. Наприклад, кома та знак рівняння (оператор присвоювання) є операціями в С; крім операції додавання `+`, є ще операція збільшити на `+=` та операція збільшення на одиницю `++`. Найчастіше вони дозволяють писати естетично гарні, але не дуже зрозумілі для починаючих програми.

Втім, ці незвичайні операції можна не використовувати, замінюючи їх традиційними.

До чотирьох звичайних арифметичних операцій додавання `+`, віднімання `-`, множення `*` та ділення `/` у Сі додана операція знаходження остачі від ділення першого цілого числа на друге, котра позначається символом відсотка `%`. Пріоритет операції обчислення остачі `%` такий же, як і у ділення або множення. Відзначимо, що операція `%` перестановочна з операцією зміни знака (унарним мінусом), наприклад, за наслідками виконання двох рядків

Операція порівняння порівнює два вирази. За наслідками виробляється логічне значення – `true` або `false` залежно від значень виразів. Приклади:

```
bool res;
int x, y;
res = (x == y); // true, якщо x дорівнює y, інакше false
res = (x == x); // завжди true
res = (2 < 1);  // завжди false
```

Операції порівняння в Сі позначаються в такий спосіб:

```
== дорівнює,          != не дорівнює,
> більше,             >= більше або дорівнює,
< менше,              <= менше або дорівнює.
```

Крім звичайних логічних операцій, у С є побітові логічні операції, які виконуються незалежно для кожного **окремого біта операндів**. Побітові операції мають наступні позначення:

```
& побітове логічне множення ("і");
| побітове логічне додавання ("або");
~ побітове логічне заперечення ("нє");
^ побітове додавання по модулю 2 (виключне "або").
```

(Необхідно пам'ятати, що логічні операції множення та додавання записуються за допомогою подвійних знаків `&&` або `||`, а побітові – за допомогою одинарних.)

В основному побітові операції застосовуються для маніпуляцій з бітовими масками. Наприклад, нехай ціле число `x` описує набір ознак деякого об'єкта, що складає із чотирьох ознак. Назвемо їх умовно `A, B, C, D`. Нехай за ознаку `A` відповідає нульовий біт слова `x` (біти у двійковому поданні числа нумеруються з права наліво, починаючи з нуля). Якщо біт дорівнює одиниці (програмісти говорять біт установлений), то вважається, що об'єкт має ознаку `A`. За ознаки `B, C, D` відповідають біти з номерами 1, 2, 3. Загальноприйнята практика полягає в тому, щоб визначити константи, відповідальні за відповідні ознаки (їх звичайно називають масками):

```
const int MASK_A = 1;
const int MASK_B = 2;
const int MASK_C = 4;
const int MASK_D = 8;
```

Ці константи містять одиницю у відповідному біті та нулі в інших бітах. Для того щоб перевірити, чи встановлений у слові `x` біт, що відповідає, приміром, ознаці `D`, використовується операція побітового логічного множення. Число `x` множиться на константу `MASK_D;` якщо результат відмінний від нуля, то біт встановлений, тобто об'єкт має ознаку `D`, якщо ні, то не встановлений. Така перевірка реалізується наступним фрагментом:

```
if ((x & MASK_D) != 0) {
    // Біт D установлений у слові x, тобто
    // об'єкт має ознаку D
    . . .
} else {
    // Об'єкт не має ознаку D
    . . .
}
```

При побітовому логічному множенні константа `MASK_D` обнулює всі біти слова `x`, крім біта `D`, тобто якби вирізує біт `D` з `x`. У двійковому поданні це виглядає так:

```
          x:   0 1 0 1 1 1 0 1 1 0 . . . 1 0 * 1 0 1
    MASK_D:    0 0 0 0 0 0 0 0 0 0 . . . 0 0 1 0 0 0
x & MASK_D:    0 0 0 0 0 0 0 0 0 0 . . . 0 0 * 0 0 0
```

Зірочкою тут позначене довільне значення біта D слова x.

Для встановлення біта D у слові x використовується операція побітового логічного додавання:

```
x = (x | MASK_D);    // Установити біт D у слові x
```

Частіше це записується з використанням операції типу "збільшити на":

```
x |= MASK_D;     // Установити біт D у слові x
```

У двійковому представленні це виглядає так:

```
x:           0101110110...10*101
MASK_D:      0000000000...001000
x | MASK_D:  0101110110...101101
```

Операція побітового заперечення "~" інвертує біти слова:

```
x:       0101110110...101101
~x:      1010001001...010010
```

Для очищення (тобто установки в нуль) біта D використовується комбінація операцій побітового заперечення та побітового логічного множення:

```
x = (x & ~MASK_D);  // Очистити біт D у слові x
```

або, застосовуючи операцію "&=" типу "помножити на":

```
x &= ~MASK_D;    // Очистити біт D у слові x
```

Тут спочатку інвертується маска, що відповідає біту D,

```
MASK_D:     0000000000...001000
~MASK_D:    1111111111...110111
```

у результаті виходять одиниці у всіх бітах, крім біта D. Потім слово x побітно помножується на інвертовану маску:

```
x:                  0101110110...10*101
~MASK_D:            1111111111...110111
x & ~MASK_D:        0101110110...100101
```

У результаті в слові x біт D обнулюється, а інші біти залишаються незмінними.

Операції зміщення застосовуються до цілочисельних змінних: двійковий код числа зміщується вправо або вліво на зазначену кількість позицій. Зміщення вправо позначається двома символами "більше" >>, зміщення вліво – двома символами "менше" <<. Приклади:

```
int x, y;
. . .
x = (y >> 3);    // Зміщення на 3 позиції вправо
y = (y << 2);    // Зміщення на 2 позиції вліво
```

При зміщенні вліво на k позицій молодші k розрядів результату встановлюються в нуль. Зміщення вліво на k позицій еквівалентно множенню на число 2^k. Зміщення вправо більше складне, воно по-різному визначається для беззнакових та знакових чисел. При зміщенні вправо беззнакового числа на k

позицій, k старших розрядів, що звільнилися, установлюються в нуль. Наприклад, у двійковому записі маємо:

```
unsigned x;

x      = 110111000...10110011
x >> 3 = 000110111000...10110
```

Зміщення вправо на k позицій відповідає цілочисельному діленню на число 2^k. При зміщенні вправо чисел зі знаком відбувається так зване "розширення знакового розряду". А саме, якщо число додатне, тобто старший, або знаковий, розряд числа дорівнює нулю, то відбувається звичайне зміщення, як і у випадку беззнакових чисел. Якщо ж число від'ємне, тобто його старший розряд дорівнює одиниці, то ті розряди, що звільнилися в результаті зміщення k старших розрядів, установлюються в одиницю. Число, таким чином, залишається від'ємним. При k=1 це відповідає діленню на 2 тільки для від'ємних чисел, не рівних -1. Для числа -1, всі біти двійкового коду якого дорівнюють одиниці, зміщення вправо не приводить до його зміни. Приклад (використовується двійковий запис):

```
int x;
x      = 110111000...10110011
x >> 3 = 111110111000...10110
```

У програмах краще не покладатися на цю особливість зміщення вправо для знакових чисел і використовувати конструкції, які свідомо однаково працюють для знакових та беззнакових чисел.

Керуючі конструкції мови дозволяють організовувати цикли та розгалуження в програмах. У С всього кілька конструкцій, причому половину з них можна не використовувати (вони реалізуються через інші).

Оператор if ("якщо") дозволяє організувати розгалуження в програмі. Він має дві форми: оператор "*якщо*" і оператор "*якщо...інакше*". Оператор "якщо" має вигляд:

```
if (умова)
    дія;
```

оператор "*якщо...інакше*" має вигляд

```
if (умова)
    дія1;
else
    дія2;
```

Як умову можна використовувати будь-який вираз логічного або цілого типу. Нагадаємо, що при використанні цілочисельного виразу значенню "істина" відповідає будь-яке ненульове значення. При виконанні оператора "якщо" спочатку обчислюється умовний вираз після if. Якщо воно дійсне, то виконується дія, якщо воно не дійсне, то нічого не відбувається. Наприклад, у наступному фрагменті в змінну m записується максимальне зі значень змінних x та y:

```
double x, y, m;
```

```
m = x;
if (y > x) m = y;
```

При виконанні оператора "*якщо...інакше*" у випадку, коли умова істинна, виконується дія, що записана після `if;` у противному випадку виконується дія після `else`. Наприклад, попередній фрагмент запишеться в такий спосіб:

```
double x, y, m;
if (x > y)
    m = x;
else
    m = y;
```

Коли треба виконати кілька дій залежно від істинності умови, необхідно використовувати фігурні дужки, поєднуючи декілька операторів у блок, наприклад:

```
double x, y, d;
if (d > 1.0) {
    x /= d;
    y /= d;  }
```

Тут змінні `x` та `y` діляться на `d` тільки в тому випадку, коли значення `d` більше одиниці.

Кілька умовних операторів типу `"якщо...інакше"` можна записувати послідовно (тобто дія після `else` можна знову записати умовний оператор). У результаті реалізується вибір з декількох можливостей. Конструкція вибору використовується в програмуванні дуже часто. Приклад: є дійсна змінна *x*, необхідно записати у дійсну змінну `y` значення функції `sign(x)`:

```
sign(x) = -1, при x < 0
sign(x) = 1,  при x > 0
sign(x) = 0,  при x = 0
```

Це робиться з використанням конструкції вибору:

```
double x, s;
if (x < 0.0) {
    s = (-1.0);
}
else if (x > 0.0) {
    s = 1.0;
}
else {
    s = 0.0;
}
```

При виконанні цього фрагмента спершу перевіряється умова `x < 0.0`. Якщо вона дійсна, то виконується оператор `s = (-1.0);` інакше перевіряється друга умова `x < 0.0`. У випадку його істинності виконується оператор `s=1.0`, інакше виконується оператор `s=0.0.` Фігурні дужки додані для поліпшення структурності тексту програми.

У кожному разі, у результаті виконання конструкції вибору виконується лише один з операторів (можливо, складених). Умови перевіряються послідовно зверху донизу. Як тільки є дійсна умова, то виконується відповідна дія і вибір закінчується.

Конструкція циклу "доки" відповідає циклу while:

```
while (умова)
    дія;
```

Цикл while називають циклом із передумовою, оскільки умова перевіряється перед виконанням тіла циклу. Цикл while виконується в такий спосіб: спочатку перевіряється умова. Якщо вона дійсна, то виконується дія. Потім знову перевіряється умова; якщо вона дійсна, то знову повторюється дія, і так нескінченно. Цикл завершується, коли умова стає помилковою. Приклад:

```
int n, p;
p = 1;
while (2*p <= n)
    p *= 2;
```

У результаті виконання цього фрагмента в змінної p буде обчислений максимальний ступінь двійки, який не переважає цілого позитивного числа n.

Якщо необхідно перервати виконання циклу, то необхідно використовувати оператор

```
break;
```

Оператор застосовується усередині тіла циклу у фігурних дужках. Приклад: потрібно знайти корінь цілочисельної функції f(x), дійсної для цілочисельних аргументів.

```
int f(int x);   // Опис прототипу функції
. . .
int x;
. . .
// Шукаємо корінь функції f(x)
x = 0;
while (true) {
    if (f(x) == 0) {
        break;  // Знайшли корінь
    }
    // Переходимо до наступного цілого значення x
    //    у порядку 0, -1, 1, -2, 2, -3, 3, ...
    if (x >= 0) {
        x = (-x - 1);
    }
    else {
        x = (-x);
    }
}
```

```
// Твердження: f(x) == 0
```

Тут використовується нескінченний цикл `"while (true)"`. Вихід із циклу здійснюється за допомогою оператора `"break"`.

Іноді потрібно пропустити виконання тіла циклу при яких-небудь значеннях змінних у циклі, переходячи до наступного набору значень і чергової ітерації. Для цього використовується оператор:

```
continue;
```

Оператор `continue`, так само, як і `break`, використовується лише в тому випадку, коли тіло циклу складається більш ніж з одного оператора та вставлено у фігурні дужки. Його варто розуміти як перехід на фігурну дужку, що закриває тіло циклу. Приклад: нехай заданий `n+1` крапка на дійсній прямій x_i, *i=0,1,…,n*; крапки x_i будуть називатися вузлами інтерполяції. Елементарний інтерполяційний багаточлен Лагранжа $L_k(x)$ – це багаточлен ступеня *n*, що приймає нульові значення у всіх вузлах x_i, крім x_k. В *k*-ом вузлі x_k багаточлен $L_k(x)$ приймає значення *1*. Багаточлен $L_k(x)$ обчислюється по наступній формулі:

$$L_k(x) = \prod_{\substack{i=1 \\ s \neq k}}^{n} \frac{(x - x_i)}{x_k - x_i}.$$

Нехай потрібно обчислити значення елементарного інтерполяційного багаточлена $L_k(x)$ при значенні $x = t$. Це робиться за допомогою наступного фрагмента програми:

```
double x[100]; // Вузли інтерполяції (не більше 100)
int n;    // Кількість вузлів інтерполяції
int k;    // Номер вузла
double t; // Крапка, у якій обчислюється значення
double L; // Значення багаточлена L_k(x) у t
int i;
L = 1.0;  // Початкове значення добутку
i = 0;
while (i <= n) {
    if (i == k) {
        ++i;      // До наступного вузла
        continue; // Пропустити k-й множник
    }
    // Обчислення добутку
    L *= (t - x[i]) / (x[k] - x[i]);
    ++i; // До наступного вузла
} // Відповідь у змінної L
```

Тут оператор `continue` використовується для того, щоб пропустити обчислення при `i = k`.

Циклу `for` має вигляд:

```
for (ініціалізація; умова продовження; ітератор)
    тіло циклу;
```

Ініціалізація виконується один раз перед першою перевіркою умови продовження та першим виконанням тіла циклу. Умова продовження перевіряється перед кожним виконанням тіла циклу. Якщо умова істинна, то виконується тіло циклу, інакше цикл завершується. Ітератор виконується після кожного виконання тіла циклу (перед наступною перевіркою умови продовження).

Оскільки умова продовження перевіряється перед виконанням тіла циклу, цикл `for` є, подібно циклу `while`, циклом із передумовою. Якщо умова продовження ніколи не виконується, то тіло циклу не виконується жодного разу, що добре як з погляду надійності програми, так і з погляду простоти та естетики (оскільки не потрібно окремо розглядати виняткові випадки).

Приклад розрахунку суми чисел масиву з використанням циклу `for`:

```
double a[100]; // Масив a містить не більше 100 эл-тів
int n;         // Реальна довжина масиву a (n <= 100)
double sum;    // Змінна для суми эл-тів масиву
sum = 0.0;
for (int i = 0; i < n; ++i) {
    sum += a[i]; // Збільшуємо суму на a[i]
}
```

Тут цілочисельна змінна `i` використовується як **змінна циклу**. В операторі ініціалізації змінної `i` присвоюється значення `0`. Умовою продовження циклу є умова `i<n`. Ітератор `++i` збільшує змінну `i` на одиницю. Таким чином, змінна `i` послідовно приймає значення `0,1,2,...,n-1`. Для кожного значення `i` виконується тіло циклу.

При вивченні керуючих конструкцій необхідно ознайомитися із логічним представлення цих конструкцій у вигляді алгоритмів з використанням графічної мови, що описана у Єдиній системі програмної документації (ЄСПД).

1.4 Варіанти індивідуальних завдань

Завдання 1

Варіант 1. Виконати виведення на екран результатів обчислення функції:

$$y = \begin{cases} 2{,}3, \textit{ при } x > 2{,}3; \\ a - \dfrac{x^2}{2}, \textit{ при } x \le 2{,}3; \end{cases}$$

$$x=\begin{cases} a^5+b, при \quad b<0; \\ a^2+\dfrac{a}{a-b}, при\ 0\le b\le 1.5,\ a\ne b; \\ a^3+ba^2+b^2, при\ 1.5<b<2.5; \\ a^4, при\ b\ge 2.5 \end{cases}$$

Змінна a приймає значення від 0 до 5 ([0, 5]) із кроком 0,5. Змінна ***b*** приймає значення від -2 до 4 ([-2, 4]) із кроком 0,1.

Варіант 2. Виконати виведення на екран результатів обчислення функції:

$$y=\begin{cases} 4{,}1, \quad при\ x>3; \\ a-\dfrac{x^3}{2}, \ при\ x\le 3; \end{cases}$$

$$x=\begin{cases} a^4+b, \quad при \quad b<0; \\ a^2+\dfrac{\sqrt{a}^5}{a-b}, при\ 0\le b\le 5,\ a\ne b; \\ a^3+4a^2+b^2, при\ 5<b<8; \\ a^5, при\ 8\le b. \end{cases}$$

Змінна ***a*** приймає значення від 0 до 5 ([0, 5]) із кроком 0,5. Змінна ***b*** приймає значення від -3 до 11 ([-3, 11]) із кроком 0,1.

Варіант 3. Виконати виведення на екран результатів обчислення функції:

$$P=3m^2-\frac{\ln(a-c)}{b^3+a};$$

$$\boldsymbol{b}= 0{,}75a^2+c^m ;$$

$$a=\begin{cases} \dfrac{\ln\left(t+d^2\right)}{\sqrt[3]{t^2d}}, \quad при\ d>t,\ t>1, \\ 2t\sin(d^2), \qquad при \quad t>1; \\ t^3d, \qquad у\ інших\ випадках. \end{cases}$$

Значення d обирається із множини $0 \div 5$ із кроком 1,1; значення t обираються із множини $0 \div 3$ із кроком 0,2; m=3. Значення змінної (c) задає користувач з клавіатури.

Варіант 4. Визначити значення функції:

$$r = \begin{cases} 2{,}35\,t^2 + l^{-tx} - a^2, & при \quad m < 0; \\ b \cdot tg\left(\dfrac{b}{2m}\right), & при \quad m \geq 0; \end{cases}$$

$$t = \begin{cases} a\sin(x-3), & при \quad b > 0; \\ \dfrac{\ln(c+b)^2}{b^2 + c - 4}, & при \quad b < 0, \quad c < 20. \end{cases}$$

Значення *m* обирається із множини – 5 ÷ 5 із кроком 0,5; a = 3; значення ***b*** обирається із множини [– 3 ÷ 3] із кроком 1; ***c*** – із множини $0 \div 50$ із кроком 10. Значення змінних (***x***) задає користувач з клавіатури.

Варіант 5. Визначити значення функції:

$$y = \begin{cases} \dfrac{a}{\ln\left(\sin^2 a\right)}, & при \quad a \neq 0 \ та \quad a \neq \pi; \\ \cos\left(x^2 + a\right), & у інших випадках; \end{cases}$$

$$x = \begin{cases} a + b - \left(3.5c + \sqrt{|b|}\right)a, & при \quad b \leq 0; \\ tgb + \dfrac{a}{b}, & при \quad b > 0.5; \\ 3.5\,ab, & в \ інших \quad випадках. \end{cases}$$

Значення a обираються із множини $0 \div 4$ із кроком 0,1, а значення $в$ – із множини – $2 \div 5$ з кроком 0,2. Значення змінної (c) задає користувач з клавіатури.

Варіант 6. Визначити значення функції:

$$b = 0.25\,a^2 + c^m;$$

$$a=\begin{cases}\ln\left(t+d^2\right) & при \quad d>t, \quad t>1, \quad d>1;\\ \sqrt{td} & при \qquad t>1;\\ t^2d & у \quad інших \quad випадках.\end{cases}$$

$$c=3m-\frac{\ln\left(a^2-d\right)}{a^3-d}.$$

Значення d обираються із множини $0\div 3$ із кроком 1, а значення t – із множини – $0\div 4$ із кроком 0,5. Значення змінних (a, m) задає користувач з клавіатури.

Варіант 7. Задані відрізки a, b, c, d. Для кожної з трійок цих відрізків, з яких можливо побудувати трикутник, визначити та вивести на екран площу цих трикутників.

Варіант 8. Задані координати вершин трикутника та координати точки у ньому. Визначити віддаль цієї точки до найближчої середини сторони трикутника. (При вирішенні задачі узято не менш 3-х варіантів координат вершин трикутника та точки у ньому).

Варіант 9. По значенню обчислити значення:

$$Z=\frac{\sqrt[3]{a}-\sqrt[6]{a^2+1}}{1-\sqrt[7]{3+a}}.$$

$$y=\sqrt[n]{Z^2}$$

Значення ***a*** обираються із множини $0\div 3$ із кроком 1,2, значення ***n*** обираються із множини [-56 ÷ 10] із кроком 5,

Варіант 10. Виконати виведення на екран результатів обчислення функції:

$$y=\begin{cases}\sqrt[25]{(a-24)^{26}-45x}, \; при \; x>2{,}3;\\ \sqrt[21]{\left(a-\dfrac{x^2}{2}\right)^{-23}}, \; при \; x\le 2{,}3;\end{cases}$$

$$x=\begin{cases} a^5+6b, & при \quad b<0; \\ 23a^2+\dfrac{45a}{a-b}, & при\ 0\le b\le 1.5,\ a\ne b; \\ a^8+2ba^2+b^8, & при\ 1.5<b<2.5; \\ a^4-9b+21ab, & при\ 2.5\le b. \end{cases}$$

Змінна a приймає значення від 0 до 5 із кроком 0,5. Змінна ***b*** приймає значення від [– 2 до 4] із кроком 0,1.

Варіант 11. Виконати виведення на екран результатів обчислення функції:

$$y=\begin{cases} \dfrac{x-78}{|x-56|+1}, при\ x>3; \\ \dfrac{\left|(x-2)^3\right|}{\ln(4-x)}, при\ x\le 3; \end{cases}$$

$$x=\begin{cases} a^7+45b+ab, & при \quad b<-10; \\ \dfrac{45a}{36b}+19a, при & -10\le b<0; \\ \sqrt[5]{8a^3+41a^2-7b^9}, при\ 5<b<8; \\ a^5, & при\ 8\le b; \\ 0, & в\ інших\ випадках. \end{cases}$$

Змінна a приймає значення від 0 до 5 із кроком 0,5. Змінна ***b*** приймає значення від [– 3…11] із кроком 0,1.

Варіант 12. Виконати виведення на екран результатів обчислення функції:

$$P=\sqrt[b]{\sqrt{|12-10m|+1}-5a-2c};$$

$$b=\sqrt[m+3]{(|3a-2|+c)^4};$$

$$a=\begin{cases}\dfrac{(p^2+5p+4)x}{\sqrt[3]{p^3x^4}} & при\ p>x,\ x>1,\\ \left(\dfrac{tgp}{tg^2p-1}\right)^x & при \quad p>1;\\ \dfrac{tg^3x}{6} & у\ інших\ випадках.\end{cases}$$

Значення ***p*** обирається із множини $0 \div 5$ із кроком 0.9; значення ***x*** обираються із множини $0 \div 3$ із кроком 0,3; ***m***=3. Значення змінної (c) задає користувач з клавіатури.

Варіант 13. Визначити значення функції:

$$\text{r}=\begin{cases}16^{2-t}+2\cdot 4^{3-x}-4^{tx}-a^4 & при \quad m<0;\\ \sin 2t\cdot\sin\left(\dfrac{3\pi}{2}+3a\right)+\cos\left(\dfrac{5\pi}{2}-2t\right) & при \quad m\ge 0;\end{cases}$$

$$t=\begin{cases}a\cos(7x-6)+\sin(a\pi) & при \quad b>0;\\ \dfrac{\ln(c+b)^2}{b^2+c-4} & при \quad b>0,\quad c<20, b\ne -c;\\ \cos^2 2b, & в\ інших\ випадках.\end{cases}$$

Значення ***m*** обирається із множини $-5 \div 5$ із кроком 0,5; ***a*** = 3; значення ***b*** обирається із множини $[-3 \div 3]$ із кроком 1; а ***c*** – із множини $0 \div 50$ із кроком 10. Значення змінних (x) задає користувач з клавіатури.

Варіант 14. Визначити значення функції:

$$y=\begin{cases}\cos 9x-\cos 7x+\cos 3a-\cos x & при \quad x\ne 0\ та \quad a\ne\pi;\\ \sin 3x+\sin 5a-\sin 4x & у\ інших\ випадках;\end{cases}$$

$$x=\begin{cases}3ab-\left(\left|3.5\cdot c-56\right|+\sqrt{|b|}\right), & при \quad b\le 0;\\ \cos^2\dfrac{a}{2}+\cos^2\dfrac{3}{2}b-\sin^2 2ab-\sin^2 4b, & при \quad b>0.5;\\ 3.5\,ab, & в\ інших \quad випадках.\end{cases}$$

Значення ***a*** обираються із множини $0 \div 5$ із кроком 0,1, а значення ***b*** – із множини – $-2 \div 5$ з кроком 0,2.

Варіант 15. Визначити значення функції:

$$b = \cos a \cdot \cos 2c - \sin\left(\frac{\pi}{4} + m\right)\sin\left(\frac{\pi}{4} + 4m\right) + \sin\left(\frac{3\pi}{4} + 4a\right)\cos\left(\frac{7\pi}{4} - 5c\right);$$

$$a = \begin{cases} \ln\left(t + d^2\right) & при \quad d > t, \quad t > 1, \quad d > 1; \\ \sqrt{td} & при \quad t > 1; \\ t^2 d & у \quad інших \quad випадках. \end{cases}$$

$$c = tg^3 a - 1 + \frac{1}{\cos^2 d} - 3ctg\left(\frac{\pi}{2} - d^3\right) \cdot 3m.$$

Значення ***d*** обираються із множини $2 \div 15$ із кроком 1, а значення ***t*** – із множини – $0 \div 5$ із кроком 0.5, ***m*** обираються із множини $-2 \div 2$ із кроком 0.41.

Варіант 16. Обчислити усі значення виразів для :

$$Z = tgx + ctgx + tg^2 x + ctg^2 x + tg^3 x + ctg^3 x,$$

$$y = \begin{cases} \dfrac{\sqrt{2} \cdot n \cdot 0{,}5^{\frac{5n}{4\sqrt{x}-10}} - 16^{\frac{1}{2n(\sqrt{x}-1)}}}{Z}, Z \neq 0; \\ 0, Z = 0. \end{cases}.$$

Значення x обираються із множини $0 \div 5$ із кроком 0.58, Значення n обираються із множини $1 \div -5$ із кроком 0.5.

Варіант 17. По значенню обчислити значення:

$$Z = 17 \cdot 2^{\sqrt{a^2+8a}} - 8 + 2 \cdot 4^{\sqrt{a^2+8a}}.$$

$$y = \sqrt[n]{Z^2}$$

Значення a обираються із множини $0 \div 5$ із кроком 0.2, значення n обираються із множини $-5 \div 10$ із кроком 2.5.

Варіант 18. Виконати виведення на екран результатів обчислення функції:

$$y=\begin{cases}4x^3-6a+1, \textit{при } x>3.6;\\ \dfrac{\sqrt[3]{5-x}}{a}, \textit{при } x\le 3.6;\end{cases}$$

$$x=\begin{cases}7\left(a+\dfrac{1}{b}\right)-2\left(b^2+\dfrac{1}{a^2}\right), & \textit{при } \quad b<0, a\neq 0;\\ a^3-b^2-\dfrac{8}{\left|a^3-b^2\right|+1}, & \textit{при } 0\le b\le 1;\\ 4\sqrt{a^2-9b}+1, & \textit{при } 1<b<2;\\ \sqrt[7]{(a-b)^3}-\sqrt[7]{(|b-a|+1)^{-3}}-\dfrac{65}{8}, & \textit{у інших випадках}.\end{cases}$$

Змінна a приймає значення від 0.1 до 5 із кроком 0,5. Змінна *b* приймає значення від [– 2, 4] із кроком 0,1.

Варіант 19. Виконати виведення на екран результатів обчислення функції:

$$y=\begin{cases}\dfrac{\sqrt{|12+x-x^2|}}{x-11}-\dfrac{\sqrt{|12+x-x^2|}}{2x-9}, \textit{при } x>12;\\ \dfrac{1+\cos(2x-2\pi)+\cos(4x+2\pi)-\cos(6x-\pi)}{\cos(2\pi-2x)+2\cos^2(2x+\pi)-1}-2\cos 2x, \textit{при } x\le 12;\end{cases}$$

$$x=\begin{cases}\sqrt[4]{\dfrac{a-b}{b+3}}+\sqrt[4]{\dfrac{b+a}{a-4}}, \textit{при} \quad b\neq -3, a\neq 4;\\ \dfrac{1-2\cos^2 2a}{2tg\left(2a-\dfrac{\pi}{4}\right)\cdot\sin^2\left(\dfrac{\pi}{4}+2b\right)}, \quad \textit{при} \quad -10\le b<0;\\ \dfrac{1}{a+b}+\dfrac{2a-b}{a-b}-\dfrac{a+b}{1}, \textit{при } b\neq\pm a;\\ \sqrt{2-a}+4b-3, \textit{в інших випадках}.\end{cases}$$

Змінна a приймає значення від 0 до 8 із кроком 0,5. Змінна $в$ приймає значення від [– 3,11] із кроком 0,2.

Варіант 20. Виконати виведення на екран результатів обчислення функції:

$$P = \sqrt{|2a-1|+3} + \sqrt{|2a+15|+9};$$

$$b = -\frac{10c^a}{\sqrt{|2m-1|+5}};$$

$$a = \begin{cases} 4\cos p \cdot tgx - 2\cos 2x - \dfrac{1}{\cos^2 p}, при\ p > x,\ x > 1 \\ \left(\dfrac{1}{2}tgx + ctg(p)\right) \cdot \left(2\sin x + tg(p)\right), \quad у\ інших\ випадках. \end{cases}$$

Значення p обирається із множини $0 \div 15$ із кроком 0.741; значення x обираються із множини $0 \div 3$ із кроком 0,3; m=3. Значення змінної (с) задає користувач з клавіатури.

Варіант 21. Визначити значення функції:

$$r = \begin{cases} \dfrac{ctg\left(\dfrac{\pi}{4}+a\right) \cdot \left(1+\cos\left(\dfrac{3\pi}{2}+2x\right)\right)}{\cos\left(2t-\dfrac{5\pi}{2}\right)+2} - ctg2\alpha, \quad при \quad m<0; \\ \sqrt{2-\sqrt{|3+a|+2}} + \sqrt{|x+4|}, \quad при \quad m \geq 0; \end{cases}$$

$$t = \begin{cases} \dfrac{b}{x-a} - \dfrac{2a}{x+a} + \dfrac{8a^2}{b^2-a^2} \quad при \quad b \neq \pm a, x \neq \pm a; \\ \dfrac{\sqrt{c^2-16}}{\sqrt{x-3}} + \sqrt{b-3} + \dfrac{5a}{\sqrt{x-3}} \quad при \quad x \geq 3, c \neq \pm 4, b > 5; \\ a^2 - 2|c| - 3b, \ у\ інших\ випадках. \end{cases}$$

Значення m обирається із множини [– 5 ÷ 5] із кроком 0,5; a = 3; значення b обирається із множини [– 3 ÷ 3] із кроком 1; а c – із множини $0 \div 50$ із кроком 10. Значення змінних (х) задає користувач з клавіатури.

Варіант 22. Визначити значення функції:

$$y = \begin{cases} \dfrac{1-2\sin^2 2a}{1-\sin 4a} - \dfrac{1+tg2a}{x} \quad при \quad x \neq 0 \ та \quad a \neq \pi; \\ \sqrt{|x^2+3x+2|} - \sqrt{|x^2-x+1|} - 1 \quad у інших\ випадках; \end{cases}$$

$$x=\begin{cases}\sqrt[3]{a+b}-\sqrt[3]{a-b}-\sqrt[6]{a^2+b^2} & при \quad b\le a;\\ \sqrt{\left|4-4a^3+b^6\right|+6}+a-\sqrt[3]{2} & при \quad b>0.5;\\ \left(a\sqrt[3]{a}-2a\sqrt[3]{b}+\sqrt[3]{a^2b^2}+\sqrt[3]{a^2b}-\sqrt[3]{ab^2}\right)\div\sqrt[3]{a^2+1} & в\ інших\ випадках.\end{cases}$$

Значення ***a*** обираються із множини $0 \div 5$ із кроком 0,1, а значення b – із множини – $-2 \div 5$ з кроком 0,2.

Варіант 23. Визначити значення функції:

$$b=\frac{c^2+1}{2}+\frac{m}{\sqrt{a^2+1}}-2.9a;$$

$$a=\begin{cases}t+d, при \quad d>t, \quad t>10, \quad d<1;\\ 2t^2-td+3d^2-7t-12d+1, при\ t>1;\\ \dfrac{t^2d}{tg6} \quad у \quad інших \quad випадках.\end{cases}$$

$$c=\frac{\left|a^2-5b+4\right|}{\left|d^2-4\right|}m.$$

Значення ***d*** обираються із множини $3 \div 15$ із кроком 1, а значення ***t*** – з множини – $0 \div 5$ із кроком 0.5, ***m*** обираються із множини $-2 \div 2$ із кроком 0.41.

Варіант 24. По значенню обчислити значення:

$$Z=\left(\frac{1}{\sqrt{a}+\sqrt{a+1}}+\frac{1}{\sqrt{a}+\sqrt{a+1}}\right)\div\left(1+\sqrt{\frac{a+1}{a-1}}\right),$$

$$y=\frac{Z^2+n}{\left|Z^2-n+1\right|+3}-\frac{n^2-Z+2}{\left|n^2-Z-2\right|+2}.$$

Значення a обираються із множини $3 \div 15$ із кроком 0.58, Значення n обираються із множини $1 \div -5$ із кроком 0.5.

Варіант 25. По значенню обчислити значення:

$$Z = \left(\frac{1+\sqrt{|1-x|}}{x^2-1+\sqrt{1+x}} + \frac{1-\sqrt{1+x}}{1+x^2-\sqrt{1+x}} \right) \div \frac{\sqrt{|1-x|}-\sqrt{1+x}}{\sqrt{x^2-1}},$$

$$y = Z^3 - n^2 - \frac{8}{|Z^3 - n^2| + 6}.$$

Значення *x* обираються із множини [3 ÷ 21] із кроком 0.62, Значення *n* обираються із множини 1 ÷ -5 із кроком 0.5.

Варіант 26. По значенню обчислити значення:

$$Z = 20\left(\frac{x-2}{x+1}\right)^2 - 5\left(\frac{x+2}{x-1}\right)^2 + 48\frac{x^2-4}{x^2-1},$$

$$y = 7^Z \left(\sqrt{2}\right)^{2n^2-6} - \left(\frac{7}{4}\right)^Z.$$

Значення *x* обираються із множини [3 ÷ 15] із кроком 0.74, Значення n обираються із множини [-5 ÷ 1] із кроком 0.5.

Завдання 2.

Варіант 1. Ввести тризначне число. Ведення повторювати до тих пір, доки буде не введено саме тризначне число. Вивести число, отримане при прочитанні його цифр справа наліво.

Варіант 2. Ввести тризначне число. Ведення повторювати до тих пір, доки буде не введено саме тризначне число. Вивести число, отримане видаленням першої цифри та переставлення її на кінцеву позицію.

Варіант 3. Ввести тризначне число. Введення повторювати до тих пір, доки буде не введено саме тризначне число. Вивести число, отримане шляхом закреслення в ньому останньої справа цифри і приписування її на початку.

Варіант 4. Ввести тризначне число. Введення повторювати до тих пір, доки буде не введено саме тризначне число. Вивести число, отримане шляхом перестановки першої та другої цифр заданого числа.

Варіант 5. Ввести тризначне число. Введення повторювати до тих пір, доки буде не введено саме тризначне число. Вивести число, отримане шляхом перестановки другої і третьої цифр заданого числа.

Варіант 6. Ввести тризначне число. Введення повторювати до тих пір, доки буде не введено саме тризначне число. Якщо в цьому числі є різні цифри, вивести три варіанта чисел, отриманих при переставленні цифр заданного числа.

Варіант 7. Визначити, чи увійде в конверт з внутрішніми розмірами a×b мм прямокутна листівка розміром c ×d мм. Для розміщення листівки в конверті необхідний зазор в 1 мм з кожного боку. Розміри конверта та лістівки вводити з екрану.

Варіант 8. Дано раціональні позитивні числа a, b, c, x, y. З'ясувати, чи пройде цеглина з ребрами a, b, c в прямокутний отвір зі сторонами x і y. Просовувати цеглу в отвір дозволяється тільки так, щоб кожне з його ребер було паралельно або перпендикулярно кожної зі сторін отвору.

Варіант 9. Ввести натуральне число n (999 < n < 10000). Введення повторювати до тих пір, доки буде введено число, що задовольняє умовам. З'ясувати, чи є число паліндромом ("перевертнем") з урахуванням чотирьох цифр, як, наприклад, числа 7777, 8338, 0330 і т. ін. (Паліндром називається число, десятковий запис якого читається однаково зліва направо і справа наліво.)

Варіант 10. Ввести натуральне число n (999 < n < 10000). Введення повторювати до тих пір, доки буде введено число, що задовольняє умовам. З'ясувати, чи вірно, що це число містить рівно три однакові цифри з урахуванням чотирьох цифр, як, наприклад, числа 3363, 4844, 0300 і т. п.

Варіант 11. Ввести натуральне число n (999 < n < 10000). Введення повторювати до тих пір, доки буде введено число, що задовольняє умовам. З'ясувати, різні чи всі чотири цифри цього числа. Наприклад, в числі 3678 всі цифри різні, в числі 0023 - ні.

Варіант 12. Ввести натуральне шестизначне число n. Введення повторювати до тих пір, доки буде введено число, що задовольняє умовам. Визначити, чи є задане число щасливим. (Щасливим називають таке шестизначне число, що сума його перших трьох цифр дорівнює сумі його останніх трьох цифр.)

Варіант 13. Ввести дійсні числа x, y. Вивести на екран номер чверті координатної площини, якій належить точка з координатами (x, y), або назву прямої (0x або 0y) якщо точка лежить на одній з вісей координат, або що точка є началом координат.

Варіант 14. Мастям гральних карт умовно присвоєні такі порядкові номери: масті "піки" - 1, масті "трефи" - 2, масті "бубни" - 3, масті "черви" - 4, а гідно карт: "валет" - 11, "дамі" - 12, "королю" - 13, "тузу" - 14 (порядкові номери карт інших достоїнств відповідають їх назвам: "шістка", "дев'ятка" і т. ін.). За заданим номером масті m (1 - 4) і номеру гідності карти k (6 - 14) визначити повну назву (масть і гідність) відповідної карти в вигляді "Дама пік", "Шістка бубон" і т. п.

Варіант 15. Дата деякого дня характеризується двома натуральними числами: m (порядковий номер місяця) і n (число). За заданим n і m визначити:

а) дату попереднього дня (прийняти, що n і m не характеризують 1 січня);

б) дату наступного дня (прийняти, що n і m не характеризують 31 грудня).

Варіант 16. Дано дійсні позитивні числа a, b, c. Якщо існує трикутник зі сторонами a, b, c, то визначити його вид (прямокутний, гострокутний або тупокутний) і особливості (рівносторонній, рівнобедрений, різносторонній).

Варіант 17. Робота світлофора для водіїв запрограмована таким чином: на початку кожної години протягом трьох хвилин горить зелений сигнал, потім протягом однієї хвилини - жовтий, протягом двох хвилин - червоний, протягом трьох хвилин - знову зелений і т. д. Дано дійсне число t, що означає час в

хвилинах, що минув з початку чергового години. Визначити, сигнал якого кольору горить для водіїв в цей момент.

Варіант 18. Відомі рік і номер місяця народження людини, а також рік і номер місяця сьогоднішнього дня (січень - 1 і т. Д.). Визначити вік людини (число повних років). У разі збігу вказаних номерів місяців вважати, що пройшов повний рік.

Варіант 19. Дано дійсні позитивні числа a, b, c, d. З'ясувати, чи можна прямокутник зі сторонами a, b вмістити всередині прямокутника зі сторонами c, d так, щоб кожна зі сторін одного прямокутника була паралельна або перпендикулярна кожній стороні другого прямокутника.

Варіант 20. Пасажир повинен був здати в камеру зберігання порожній чемодан в формі паралелепіпеда розмірами a1, a2 і a3 см і коробку розмірами b1, b2 і b3 см. Оплачувати потрібно кожен предмет, що розміщується. Визначити, чи зможе пасажир заощадити на оплаті, помістивши коробку в чемодан так, що сторони валізи і коробки будуть паралельні або перпендикулярні один одному.

Варіант 21. Дано ціле число k (від 1 до 365). Ввести його з екрану, запитувати введення до тих пір, поки не буде правильно. Визначити, яким буде k-й день року: суботою, неділею або робочим днем, якщо 1 січня у цьому році – вівторок.

Варіант 22. Ввести тризначне число. Введення повторювати до тих пір, доки буде не введено саме тризначне число. Визначити:

а) чи є сума його цифр двозначним числом;

б) чи є добуток його цифр тризначним числом.

Варіант 23. Ввести тризначне число. Введення повторювати до тих пір, доки буде не введено саме тризначне число. Визначити:

а) чи більше числа добуток його цифр;

б) чи кратна п'яти сума його цифр;

Варіант 24. Ввести тризначне число. Введення повторювати до тих пір, доки буде не введено саме тризначне число. Визначити:

а) чи кратна сума його цифр числу.

б) чи є в записі числа однакові цифри.

Варіант 25. Рік є високосним, якщо його номер кратний 4, однак з кратних 100 високосними є лише кратні 400, наприклад, 1700, 1800 і 1900 - невисокосні роки, 2000 - високосний. Ввести натуральне число. Введення повторювати до тих пір, доки буде введено рік до поточного. Визначити, чи є високосним рік з таким номером.

ФОРМА ТИТУЛЬНОГО АРКУШУ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра системотехніки

Дисципліна: «Алгоритмізація та програмування»

ЛАБОРАТОРНА РОБОТА № 1

«ВИВЧЕННЯ МОЖЛИВОСТЕЙ МОВИ С++ ПРИ РОБОТІ З ФУНКЦІЯМИ»

Виконав:
ст. гр. ІТУ-21-1
ПІБ

Прийняв:

з оцінкою «_____»
«____»_____20___р.

Харків 20___

СКЛАД РОЗДІЛІВ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

Звіт з лабораторної роботи повинен складатися з таких розділів:

1. Титульний аркуш.
1. **Мета роботи.**
2. **Постановка задачі.** Оформлюється у відповідності із індивідуальним завданням до відповідної лабораторної роботи. **Формули оформлюються з використання інструментів редагування формул (MS Equation, MathType тощо).**
3. **Хід роботи.** До складу розділу повинні входити:
   - опис вибраного або розробленого методу розв'язання задачі,
   - опис розробленого алгоритму вирішення задачі,
   - схема алгоритму,
   - текст розробленої програми (для отримання високого балу рекомендується виконати перше завдання двома способами за допомогою циклів з параметром та циклів з передумовою (постумовою),
   - контрольні приклади для перевірки та налагодження програми.
4. **Виконання експериментів по налагодженню програми.** До складу розділу повинні входити: результати трансляції та налагодження програми у відповідності із контрольними прикладами. Завершальною частиною цього розділу повинні бути результати виконання налагодження програми.
5. **Висновки.** Повинні включати висновки по результатах виконання лабораторної роботи з аналізом вибраного методу вирішення задачі та розробленого алгоритму, процесу налагодження програми, особливостей реалізації програми з використанням мови C++ та інструментального засобу Microsoft Visual Studio (або іншої IDE, програмного середовища).

Базовые функции

Имя	Описание
`abs`	Возвращает абсолютную величину целого числа
`acos`	арккосинус
`asin`	арксинус
`atan`	арктангенс
`atan2`	арктангенс с двумя параметрами
`ceil`	округление до ближайшего большего целого числа
`cos`	косинус
`random`	выводит случайное число от 0 до аргумента функции.
`exp`	вычисление экспоненты
`fabs`	абсолютная величина (числа с плавающей точкой)
`floor`	округление до ближайшего меньшего целого числа
`fmod`	вычисление остатка от деления нацело для чисел с плавающей точкой
`frexp`	разбивает число с плавающей точкой на мантиссу и показатель степени.
`ldexp`	умножение числа с плавающей точкой на целую степень двух
`log`	натуральный логарифм
`log10`	логарифм по основанию 10
`modf(x,p)`	извлекает целую и дробную части (с учетом знака) из числа с плавающей точкой
`pow(x,y)`	результат возведения *x* в степень *y*, x^y
`sin`	синус
`sinh`	гиперболический синус
`sqrt`	квадратный корень
`tan`	тангенс
`tanh`	гиперболический тангенс

Функции стандарта C++

Имя	Описание
`acosh`	гиперболический арккосинус
`asinh`	гиперболический арксинус
`atanh`	гиперболический арктангенс
`cbrt`	кубический корень
`copysign(x,y)`	возвращает величину, абсолютное значение которой равно *x*, но знак которой соответствует знаку *y*
`erf`	функция ошибок
`erfc`	дополнительная функция ошибок
`exp2(x)`	значение числа 2, возведённого в степень *x*, 2^x
`expm1(x)`	значение функции $e^x - 1$
`fdim(x,y)`	вычисление положительной разницы между *x* и *y*, fmax(*x*−*y*, 0)
`fma(x,y,z)`	значение функции (*x* * *y*) + *z* (см. FMA)
`fmax(x,y)`	наибольшее значение среди *x* и *y*
`fmin(x,y)`	наименьшее значение среди *x* и *y*
`hypot(x,y)`	гипотенуза, sqrt($x^2 + y^2$)
`ilogb`	экспонента числа с плавающей точкой, конвертированная в `int`
`lgamma`	натуральный логарифм абсолютного значения гамма-функции
`llrint`	округление до ближайшего целого (возвращает `long long`)
`lrint`	округление до ближайшего целого (возвращает `long`)
`llround`	округление до ближайшего целого в направлении от нуля (возвращает `long long`)
`lround`	округление до ближайшего целого в направлении от нуля (возвращает `long`)
`log1p(x)`	натуральный логарифм 1 + *x*
`log2`	логарифм по основанию 2

`logb`	целочисленная часть логарифма x по основанию 2
`nan(s)`	возвращает нечисловое значение 'Not a Number'
`nearbyint`	округление аргумента до целого значения в формате числа с плавающей точкой
`nextafter(x,y)`	следующий ближайшее представимое для *x* (по направлению к *y*)
`nexttoward(x,y)`	то же, что и `nextafter`, но *y* имеет тип `long double`
`remainder(x,y)`	вычисляет остаток от деления согласно стандарту IEC 60559
`remquo(x,y,p)`	то же, что и `remainder`, но сохраняет коэффициент по указателю *p* (как `int`)
`rint`	округление до целого (возвращает `int`) с вызовом ошибки `inexact`, если результат отличается от аргумента.
`round`	округление до целого (возвращает `double`)
`scalbln(x,n)`	*x* * `FLT_RADIX`^{n} (*n* is `long`)
`scalbn(x,n)`	*x* * `FLT_RADIX`^{n} (*n* is `int`)
`tgamma`	гамма-функция
`trunc`	отбрасывание дробной части