**Лекція № 5. Поняття вхідного та вихідного потоку, найпростіші математичні функції.**

**Арифметичні операції**

**Операції, операнди, вирази**

Обчислювальні дії в програмуванні називаються **операціями**. Операції застосовуються до **операндів**, тобто значень. Застосування операцій до значень описують у вигляді **виразу** (*expression*). Послідовність застосування операцій називається **обчисленням** виразу й має результатом **значення виразу**.

Операції у виразі позначаються знаками (**операторами**), а значення – константами та іменами змінних. У виразі також можуть бути дужки, що визначають порядок застосування операцій. Найпростішими виразами є ті, що не містять операцій, тобто константи та імена змінних.

**Приклад 1**.

* Вираз **2+2** означає: додаються 2 та 2 і значенням виразу є 4;
* вираз **2\*radius** – 2 множиться на значення змінної **radius** і значенням виразу є подвоєне значення цієї змінної;
* **1+2\*3** – множаться 2 та 3, отриманий добуток 6 додається до 1 ізначенням є 7;
* **(1+2)\*3** – додаються 1 і 2, їх сума 3 множиться на 3 та значенням є 9.

Два останні вирази демонструють, як дужки впливають на порядок операцій. Вираз має подвійну семантику – послідовність операцій з операндами, а також значення, що є результатом цієї послідовності.

Операція з одним або кількома значеннями, результатом якої є число, називається **арифметичною**. Спочатку розглянемо тільки деякі з багатьох арифметичних операцій мови С++.

Операції **додавання**, **віднімання**, **множення** й **ділення** мають знаки відповідно **+, -, \*, /**. Результатом операції з *цілими* числами є *ціле* число, з *дійсними* – *дійсне*. Наприклад, значенням виразу **4/2** є ціле число **2**, а виразу **4.0/2.0** – дійсне **2.0**.

Знак **–** позначає як двомісну операцію віднімання, так і одномісну операцію "мінус": **-32768**, **-(2+3)**. Знак **+** також може позначати одномісну операцію.

Результатом ділення **/** цілих чисел є *ціла частка* від ділення з остачею, наприклад, вираз **7/3** має значення **2**. *Цілу остачу* від ділення обчислює операція **%**: значенням виразу **7%3** є **1**. Зауважимо: знак остачі збігається зі знаком діленого, наприклад, обидва вирази **-7%3** та **-7%3** мають значення **-1**, а вираз **-7%-3** – значенняº **1**.

Результатом ділення дійсних чисел є число в його дійсному зображенні, наприклад, значенням виразу **7.0/3.0** є деяке наближення до числа **2.33…**, а виразу **6.0/3.0** – число **2.0**. Операція **%** до дійсних чисел незастосовна.

Виконання операції **/** або **%** з дільником 0 призводить до аварійного завершення програми.

Одномісна операція обчислює цілу *кількість байтів*, зайнятих її операндом (дані типу **char** займають 1 байт, типу **int** – 4 байти, **double** – 8). Отже, під час виконання інструкції

**cout<< sizeof 'A' << ' ' << sizeof 1 << ' ' << sizeof 0.0 << endl;**

отримаємо **1 4 8**.

**Старшинство операторів і порядок виконання операцій**

Мова С++ в основному відповідає угодам математики про порядок застосування операцій у виразах. Це дозволяє не записувати зайві дужки, наприклад, **1-2\*3** означає те саме, що й **1-(2\*3)**. На порядок обчислення виразу за відсутності дужок впливає **старшинство** (*precedence*), або **пріоритет**, операторів: якщо поруч із позначенням операнда записано два оператори, то спочатку виконується операція, що відповідає старшому оператору (з вищим пріоритетом). Наприклад, пріоритети **\*** та **/** однакові й вищі за **+** і **-**. Одномісні оператори старші за двомісні, а двомісні **\***, **/**, **%**, -, **+**, старші за всі інші двомісні, у тому числі присвоювання.

Окрім пріоритетів, оператори мають властивості право- або лівобічного зв'язування. У мові С++ усі двомісні оператори, окрім присвоювань, мають властивість **лівобічного зв'язування**: якщо ліворуч і праворуч від позначення операнда записано знаки операцій з однаковим старшинством, то операнд зв'язується з оператором, указаним ліворуч (ця операція застосовується спочатку).

**Приклади 2.**

1. Значенням виразу **4+7/5** є **5**, оскільки спочатку обчислюється **7/5** із результатом **1**, а потім **4+1** із результатом **5**. Значенням виразу **(4+7)/5** є **2**, оскільки спочатку обчислюється операція в дужках **(4+7)** – її значення 11, а потім **11/5** із результатом **2**.

2. У виразі **sizeof 2.0+4** обчислюється **sizeof** із результатом **8**, потім додається **4**.

3. Значення виразу **4-3-2** дорівнює **-1**, оскільки спочатку обчислюється **4-3**, тобто **1**, а потім **1-2**; у виразі **2\*7%8** спочатку обчислюється **2\*7** (це **14**), потім **14%8**, тобто **6**.

4. Нехай дійсні змінні **a**, **b**, **c** зображують коефіцієнти квадратного рівняння *ax*2+*bx*+*º*=º0. Дискримінант рівняння визначається виразом **b\*b–4\*a\*c**. Присвоїмо його дійсній змінній **d**: **d=b\*b–4\*a\*c**.

Пріоритети операторів дозволяють не записувати зайві дужки, але зловживати цим не слід. Інколи необов'язкова пара дужок значно підвищує зрозумілість запису. Наприклад, у виразі **sizeof 2.0+4** пробіл між **sizeof** та **2.0** провокує людину спочатку (помилково) обчислити **2.0+4**, а потім **sizeof**. Проте **sizeof(2.0)+4** є очевидним.

**Збільшення та зменшення**

У циклічних обчисленнях дуже часто використовуються присвоювання вигляду **x=x+1** та **x=x-1**. Їх можна задати в скороченій формі за допомогою одномісних операторів **збільшення** (інкременту) **++** і **зменшення** (декременту)º**--**. Ці оператори (і відповідні операції) мають **префіксну (++x, --x**) і **постфіксну** (**x++, x--**) **форми**.

Вираз із постфіксним оператором **x++** або **x--** змінює значення змінної **x** на **1**, але значенням самого виразу є значення **x** *перед зміною*. Вираз із префіксним оператором **++x** або **--x** теж змінює **x** на **1**, але значенням виразу є значення **x**, отримане *після зміни*. Ці відмінності виявляються, коли оператори **++** та **--** застосовуються всередині виразів.

Операції **++** та **--** виконуються швидше ніж відповідні присвоювання вигляду **x=x+1** та **x=x-1**, тому рекомендується використовувати саме їх. Операції **++** та **--** застосовні до змінних будь-якого з базових типів, хоча найчастіше їх використовують із цілими змінними.

Скрізь, де немає необхідності використовувати старе значення змінної, рекомендується з виразів вигляду **n++** та **++n** вибирати **++n**, оскільки він виконується швидше й простіше.

Спосіб і порядок обчислення виразу залежить від компілятора, тому краще записувати операції збільшення або зменшення в окремих виразах або інструкціях, а не у складі інших виразів. Наприклад, значення виразів **(n++)\*(n++)** та **(++n)\*(++n)** у різних системах програмування навіть можуть відрізнятися. Гарантовано лише те, що до значення змінної **n** двічі додається 1.

**Бібліотечні математичні функції та константи**

Деякі операції з числами позначають **викликами функцій**, тобто у вигляді ***f*(…)**, де ***f*** позначає певне ім'я. Розглянемо дві функції, означені в усіх реалізаціях мови С++. Для використання цих функцій у програмі необхідно підключити модуль **cmath**:

**#include <cmath>**

Одномісна функція **sqrt** обчислює квадратний корінь свого невід'ємного **дійсного** операнда. Значенням виразу **sqrt(2.0)** є приблизно **1.41421**, а **sqrt(4.0)** –значення **2.0**. До цілих чисел функція *незастосовна*.

Двомісна функція **pow** обчислює дійсний ступінь, основою якого є перший операнд, показником – другий. Наприклад, значенням виразу **pow(2.0, 3)** є **8.0**, виразу **pow(2, 0.5)** – приблизно **1.41421**. Результатом функції **pow** завжди є дійсне значення.

Функція **log** обчислює натуральний логарифм свого додатного дійсного аргументу, функція **log10** – десятковий логарифм.

Застосування функцій до цілих аргументів є помилковим.

Функція **fabs** обчислює дійсне значення |*x*| за дійсним аргументом *x*. Функція **abs** із бібліотеки **cstdlib** обчислює ціле значення |*x*| за цілим аргументом *x*; якщо аргумент дійсний; обчислене значення може відрізнятися від математичного.

**Приклади 3.**

1. Корінь із невід'ємного дискримінанта квадратного рівняння з дійсними коефіцієнтами **a, b, c** можна обчислити виразом **sqrt(b\*b–4\*a\*c)**, а дійсні корені рівняння – виразами**(-b+sqrt(b\*b–4\*a\*c))/(2\*a)** та **(-b+sqrt(b\*b–4\*a\*c))/(2\*a)**.

Дужки в знаменнику обов'язкові. Якщо їх не записати, то відбудеться не ділення, а множення на **a**.

2. Вираз **pow(b\*b–4\*a\*c,0.5)** позначає обчислення квадратного кореня з **b\*b–4\*a\*c**, вираз **pow(b,1.0/3.0)** – обчислення кубічного кореня з **b**, а обидва вирази **pow(2.0, 5)** та **pow(2, 5.0)** – піднесення дійсного числа **2.0** до степеня **5**. Зверніть увагу: вираз **pow(2, 5)** із двома цілими аргументами є помилковим.

3. Значенням **log10(2.0)** є (наближено) 0.30103, значенням **log(1)** – дійсне 0.

4. Значенням **fabs(-2.0)** є дійсне **2.0**, значенням **abs(-2)** – ціле **2**.

У стандарті мови C++ відсутні математичні константи, зокрема ті, що позначають числа π=3.141593… та *e* =2.7182818….

Натомість у бібліотеці **cmath** означено константи з іменами **M\_PI** (число π), **M\_PI\_2** (π /2), **M\_PI\_4** (π /4), **M\_1\_PI** (1/ π), **M\_E** (число *e*), **M\_LN2** (ln 2), **M\_LN10** (ln 10) і деякі інші. Щоб користуватися ними, необхідно перед підключенням бібліотеки **cmath** записати директиву **#define \_USE\_MATH\_DEFINES** (define – означити).

**Приклад 4**. Надана програма виводить значення математичних констант π та *e*.

**#include <iostream>**

**#define \_USE\_MATH\_DEFINES**

**#include <cmath>**

**using namespace std;**

**int main() {**

**cout<<"pi="<<M\_PI<<endl;**

**cout<<"e="<<M\_E<<endl;**

**cout<<endl;system("pause"); return 0;**

**}**

Бібліотеки систем програмування мовою С++ містять різноманітні константи й численні підпрограми, що реалізують математичні та інші функції. Зауважимо: склад бібліотек у різних середовищах може бути різним, тому вичерпну інформацію про вміст бібліотек може дати лише довідка в конкретному середовищі або самі бібліотечні файли. У деяких версіях мови C++ ці константи замінено відповідними бібліотечними функціями.

**Функція** в мові **С++** – це частина програми, оформлена спеціальним чином. Якщо програма описує дії з розв'язання деякої задачі, то функція описує дії з розв'язання деякої частини цієї задачі, тобто підзадачі.

**Математичні функції**

Якщо треба використовувати у програмі математичні функції, слід долучити бібліотеку, яка містить ці функції, тобто увести директиву **#include <math. h>**

|  |  |
| --- | --- |
| **acos** | арккосинус |
| **asin** | арксинус |
| **atan** | арктангенс |
| **ceil** | округлення до найближчого більшого цілого числа |
| **cos** | косинус |
| **exp** | показникова функція |
| **abs (fabs)** | модуль цілого (дійсного)  числа |
| **log** | натуральний логарифм |
| **log10** | десятковий логарифм |
| **pow(x,y)** | вираховує значення x в степені у |
| **sin** | синус |
| **sqrt** | квадратний корінь |
| **tan** | тангенс |

**Основи введення-виведення інформації**

Вирішення навіть найпростішої задачі на комп’ютері не обходиться без операцій введення-виведення інформації. Введення даних - це передача інформації ззовні в оперативну пам‘ять (далі – ОП) із зовнішнього носія; виведення даних - зворотний процес, коли дані після обробки передаються з ОП на зовнішній носій. Зовнішнім носієм може служити дисплей, друкований пристрій, гнучкий або жорсткий диск і т.ін. Передача даних програмі та виведення результатів програми є необхідним елементом програми.

Програми можуть отримувати вхідні дані декількома способами: із стандартного вхідного файлу, пов'язаного з клавіатурою, або із дискового файлу. Аналогічний поділ існує і при виведенні даних: результати роботи програми за замовчуванням направляються на екран дисплея (у стандартний вихідний файл), але можна перенаправляти їх і на інший носій інформації, наприклад, на дисковий файл, на контролер будь-якого периферійного пристроюкомп'ютера тощо.

У сучасних мовах програмування основним поняттям введення та виведення даних є *потік* — послідовність символів або інших даних. У програмі потік зображує фізичний файл на зовнішньому носії даних (диску, клавіатурі або екрані монітора), тобто фізичний файл «видно в програмі» як потік даних. Операції обміну даних з файлом представлено в програмі як операції добування даних з потоку або дописування їх до нього. Cтандартним файлам введення та виведення у таких мовах відповідають стандартний потік введення (**cin** - в C++) та стандартний потік виведення (**cout** - в C++).

Алгоритмічні мови програмування використовують концепцію поелементного введення-виведення даних. Зазвичай, введення інформації з клавіатури супроводжується "ехо-сигналом" - на екрані дисплея з'являється зображення символів, що вводяться. В деяких системах програмування існують засоби, що дозволяють відключити таке ехо. Виведення даних організується, починаючи з позиції розміщення курсору. При цьому пробіли між даними, що виводяться, автоматично не вставляються - їх необхідно враховувати самим.

За допомогою відповідних засобів можна вводити або виводити дані тільки певних типів.

Для організації у мові С++ введення та виведення даних використовуються стандартні бібліотечні функції. Бібліотеки С++ підтримують два основних способи введення/виведення: потокове введення/виведення (заголовний файл fstream) та форматоване введення/виведення за допомогою функцій (заголовний файл stdio.h).

Потокове введення / виведення.

У мові C++ дії, що пов’язані з операціями введення і виведення, виконуються за допомогою функцій бібліотек. Функції ведення і виведення бібліотек мови дозволяють читати дані з файлів та пристроїв і писати дані у файли і на пристрої.

Бібліотека мови C++ підтримує три рівня введення-виведення даних:

* введення-виведення потоку;
* введення-виведення нижнього рівня;
* введення-виведення для консолі і порту.

При введенні-виведенні потоку всі дані розглядаються як потік окремих байтів. Для користувача потік — це файл на диску або фізичний пристрій, наприклад, дисплей чи клавіатура, або пристрій для друку, з якого чи на який направляється потік даних. Операції введення-виведення для потоку дозволяють обробляти дані різних розмірів і форматів від одиночного символу до великих структур даних..

За замовчуванням стандартні введення і виведення повідомлень про помилки відносяться до консолі користувача (клавіатури та екрана). Це означає, що завжди, коли програма очікує введення зі стандартного потоку, дані повинні надходити з клавіатури, а якщо програма виводить дані — то на екран.

Потокове виведення виконується за допомогою вихідного потоку cout та операції вставки у вихідний потік ”<<”. Стандартна операція ”<<” для операцій виведення перевизначається, тобто для неї визначаються нові можливості. Потокове введення аналогічне виведенню, але використовує перевизначену операцію вставки з вхідного потоку ”>>” для потоку введення cin.

Виконуючи операцію введення з клавіатури, комп’ютер тимчасово зупиняється і очікує на введення значення для змінної. У відповідь слід на клавіатурі набрати деяку послідовність символів, що зображує значення (ці символи з’являться на екрані). Уведені символи запам’ятовуються у буфері та передаються функціям введення тільки після натиснення клавіші Enter.

*Буфер* — це область пам’яті для тимчасового зберігання даних. Максимальний обсяг буфера становить 128 символів (байтів). Завдяки наявності буфера можливе редагування даних під час їх введення.

У мові C++ існує декілька бібліотек, які містять засоби введення-виведення, наприклад:**stdio.h, iostream.h.** Найчастіше застосовують потокове введення-виведення даних, операції якого включені до складу класів **istream** або **iostream**. Доступ до бібліотеки цих класів здійснюється за допомогою використання у програмі директиви компілятора **#include <iostream.h>** в пізніших версіях **#include <iostream>**.

**Для потокового введення даних** вказується операція «**>>**» («читати з»). Це перевантажена операція, визначена для всіх простих типів і покажчика на **char**. Стандартним потоком введення є **cin**.

Формат запису операції введення має вигляд:

**сin [>> values];**

де ***values*** — змінна.

Так, для введення значень змінних **х і у** можна записати:

**сin >> х >> у;**

Кожна операція «>>» передбачає введення одного значення. При такому введенні даних необхідно дотримуватись конкретних вимог:

* для послідовного введення декількох чисел їх слід розділяти символом пропуску (« ») або **Enter** (дані типу **char** роздiляти пропуском необов’язково);
* якщо послідовно вводиться символ і число (або навпаки), пропуск треба записувати тільки в тому випадку, коли символ (типу **char**) є цифрою;
* потік введення ігнорує пропуски;
* для введення великої кількості даних одним оператором їх можна розташовувати в декількох рядках (використовуючи Enter);
* операція введення з потоку припиняє свою роботу тоді, коли всі включені до нього змінні одержують значення. Наприклад, для операції введення **х і у**, що вказана вище, можна ввести значення **х та у** таким чином:

**2.345 789**  
або

**2.345**  
**789.**

Оскільки в цьому прикладі пропуск є роздільником між значеннями, що вводяться, то при введенні рядків, котрі мiстять пропуски у своєму складі, цей оператор не використовується. У такому випадку треба застосовувати функції **getline(), get()** тощо (це буде розглядатися окремо в подальших темах). У мові C++ бажано здійснювати потокові введення-виведення даних.

За стандартних налаштувань запис числа сприймається як десятковий. Якщо натиснути Enter, не набравши нічого, окрім пропусків, то комп’ютер і надалі чекатиме. В інструкції введення можна записати кілька імен змінних - кожне після «свого» знака ”>>”. При виконанні такої інструкції треба набрати на клавіатурі відповідну кількість вхідних констант, відокремивши їх одним або кількома порожніми символами.

**Для потокового виведення даних** необхідна операція **«<<»** («записати в»), що використовується разом з ім’ям вихідного потоку **cout**. Наприклад, вираз

**cout << х;**

означає виведення значення змінної **х** (або запис у потік). Ця операція вибирає необхідну функцію перетворення даних у потік байтів.

Формат запису операції виведення представляється як:

**cout << data [<< data1];**,

де **data, data1** — це змінні, константи, вирази тощо.

Потокова операція виведення може мати вигляд:

**cout << “у =” << x + а – sin(x) << “\n“;.**

Застосовуючи логічні операції, вирази треба брати в дужки:

**cout << “р =” << (а && b || с) << “\n“;.**

Символ переведення на наступний рядок записується як рядкова константа, тобто **“\n”**, інакше він розглядається не як символ керуючої послідовності, а як число 10 (код символу).

Слід пам’ятати, що ***при виведенні даних з використанням «cout <<» не виконується автоматичний перехід на наступний рядок, для реалізації такого переходу застосовується так званий знак переведення рядка*** ***“\n” або операція*** ***endl.*** Тобто, вивести рядкову константу можна, наприклад, так:

**cout <<** **“**Bad news has wings **\n“;**

**або   cout <<** **“**Bad news has wings **” << endl;**.

Операція «помістити в потік» (”<<”) має асоціативність зліва направо:

(((cout<<"47+ 53 =")<<(47+53))<<endl)

Тобто, багаторазові операції ”<<” виконуються у тій послідовності, у якій вони записані. Такий спосіб об’єднання операцій ”<<” можливий, оскільки дана операція повертає посилання на об’єкт свого лівого операнда (на об’єкт cout). Отже, саме лівий вираз в круглих дужках (cout << "47 + 53 =")виводить заданий рядок символів і повертає посилання на cout*.* Тому середній вираз в круглих дужках має виконуватися, тобто виводити ціле значення 100 і повертає посилання на cout*,* наприклад: (cout<<(47+53)). Потім виконується самий правий вираз у круглих дужках, наприклад: cout << endl*,* який переводить рядок, скидає cout і повертає посилання на cout. Аналогічно, багаторазовою є і операції ”>>”.

Для керування потоковим введенням/виведенням можуть використовуватися маніпулятори потоків, які дозволяють міняти режими виконання операцій введення-виведення. Наприклад, маніпулятор **hex** задає форматування із шістнадцятковим перетворенням, **endl** - вставку символа нового рядка і очищення потоку.

Стандартні засоби потокового введення/виведення описані у бібліотеці iostream (iostream.h), маніпулятори потоків - у бібліотеці iomanip.h. Щоб мати змогу їх використати у програмі, відповідні бібліотеки слід підключити до програми за допомогою директиви #include. Наприклад,

#include <iostream.h>

int main()

{ double r,s;

const float pi=3.14;

cout<< "\nr="; cin >> r;

s=pi\*r\*r;

cout<< "\narea of a circle: " << s << endl;

system("pause");

return 0;

}

Форматоване введення / виведення.Здійснюєтьсяфункціями:

* **printf** - виводить аргументи у стандартний потік stdout у відповідності із заданим форматом;
* **scanf** - вводить дані із стандартного потоку вводу stdin у відповідності із заданим форматом, записуючи їх у змінні, адреси яких задаються аргументами.

Щоб зв’язати програму користувача зі стандартною бібліотекою, де знаходяться ці функції, необхідно на початку програми включити заголовний файл **stdio.h**.

*Формат виклику функції printf*:

**printf("форматний\_рядок", [список\_аргументів])**

Де *список\_аргумен*тів = аргумент {, аргумент}

*форматний\_рядок* = “”” ([літерал] | % [маркер][ширина][.точність] тип)

{( [літерал] | % [маркер][ширина][.точність] тип}”””

Форматний рядок завжди береться в подвійні лапки. Символ "%" є символом початку задання формату введення чергового аргумента. При описі формату виведення також використовуються:

*маркер* – специфікатор вирівнювання виведених знаків, пропусків, десяткових точок, вісімкових і шістнадцяткових префіксів (наприклад, вирівнювання результату перетворення по лівій межі, обов'язкове зображення знака числа тощо);

ширина– загальна ширина поля виведення;

*точність* – максимальне число цифр дробової частини числа, які будуть виведені після коми;

*тип* – специфікатор типу аргумента (наприклад, d – ціле десяткове число, f – дійсне число).

Функція printf переводить дані з внутрішнього коду в символьне представлення відповідно до форматного рядка і виводить отримані символи на екран. Форматний рядок може включати довільний текст, керуючі символи та специфікації перетворення даних. Список аргументів є необов'язковим параметром даної функції. Дану функцію можна використовувати для виводу будь-якої комбінації символів, цілих та дійсних чисел, тощо.

**Функція printf** може використовуватися, наприклад, для виведення повідомлення на екран:

**printf ("**Enter the source data **\n");**

Для звертання до функції використовуються параметри, якi розташовані у круглих дужках. Найчастіше функція **printf** реалізується для виведення значень змінних. Першим аргументом у звертанні до функції ставиться рядок форматів (береться в лапки), а наступними, якщо вони є, — об’єкти, що виводяться.

Рядок форматів може включати звичайні символи, які копіюються при виведенні, і специфікації перетворення, що починаються із символу«% », за специфікаціями йде символ перетворення. Кожна специфікація перетворення відповідає одному з аргументів, що йдуть за форматним рядком, і між ними встановлюється взаємно однозначна відповідність, наприклад:

**printf (“Values a, b, c are equal: %d %d. %d \n”, а, b, с);**

тут літера **d** у специфікації перетворення вказує, що значення аргументу має бути представлено як десяткове ціле число.

Список форматних кодів має таку форму запису:

**% [прапорець] [довжина] [точність] [f | n] [h | l] тип ,**

де **прапорець** — символ, що керує вирівнюванням виведення і виведенням пропусків, десяткової крапки, ознак чисел вісімкової і шістнадцяткової систем числення. **Прапорець** може задаватися одним із символів:

**«-»** — вирівнювання вліво усередині заданого поля;

**«+»** — виведення знака числа;

**« »** (пропуск) — приєднання пропуску до виведеного числа, якщо число є додатним і має тип зі знаком;

**«#»** — виводиться ідентифікатор системи числення для цілих: **0** — для вісімкових чисел, **0х** чи **0Х** — для шістнадцяткових чисел;

**довжина** — визначає мінімальну кількість виведених символiв, якщо довжина більше виведеної кількості символів, то виведене значення доповнюється пропусками, у випадку, коли довжина менше кількості символів у виведеному значенні або вона не задана, виводяться всі символи значення (відповідно до поля точність, якщо воно є);

**точність** — задається цілим числом після крапки і визначає кількість виведених символів, кількість знаків після крапки; на відміну від поля довжини поле точність може привести до «зрізання» виведених даних.

Параметри **f, n, h, l** і **тип** списку форматних кодів за змістом аналогічні раніше описаним для функції **scanf**.

Виведення результатів з використанням форматних кодів функції **printf** може мати вигляд:

**printf (” % 3.0 f % 6.1 f \ n ”, х, у);**

***Приклад 5.*** Обчислити значення функції **у = ах2 – sinx**, якщо **а =  10,5; х є [-1; 2]; hx = 0,5**.

**#include <stdio.h>**

**#include <math.h>**

**#include <conio.h>**

**main( )**

**{ float x, y, a(10.5);**

**printf ("\t Vivod rezultata\n");**

**for (x = -1; x <= 2; x += 0.5)**

**{** **у = a \* pow(x,2) - sin(x);     //у = a\*x\*x - sin(x);**

**printf (" \t x = % 4.1f    у = % 6.3f \n", x, y);**

**}**

**getch ();** *//затримка екрана*

**}**

Результати обчислення:  
**Vivod rezultata**  
**x = -1.0 у =11.342**  
**х = -0.5 у = 3.104**  
**х = 0.0  у = 0.000**  
**х = 0.5  у = 2.146**  
**х = 1.0  у = 9.659**  
**х = 1.5  у = 22.628**  
**х = 2.0  у = 41.091.**

Формат виклику функції scanf:

**Функція scanf**, що забезпечує форматоване введення даних, має змінне число параметрів, при цьому перед відповідним параметром ставиться знак **«&»** — символ взяття адреси змінної. Наприклад, **&х1** означає адресу змінної **х1**, а не значення, яке ця змінна має в даний момент. Рядок форматів функції **scanf** вказує, які дані очікуються на вході. Якщо функція зустрічає у форматному рядку знак **« % »**, за яким розташований знак перетворення, то на вході будуть пропускатися символи, доки не з’явиться деякий непорожній символ.

Форма запису функції **scanf** має вигляд:

**scanf ("рядок форматних кодів", список імен змінних);**

Рядок форматних кодів являє собою таку структуру запису:

**%[ \* ][довжина][f|n][h|l] тип**

де **«%»** — ознака початку форматного коду. Якщо за символом **«% »** йде символ, що не є символом керування форматом, то він розглядається як звичайна послідовність символів. При цьому наступні за ним символи (до наступного символу **«%»**) також вважаються просто символами; якщо за символом **«%»** йде символ **«\*»**, то присвоювання наступного вхідного поля приглушується, поле читається, але не зберігається;

**довжина** — позитивне десяткове ціле число, яке задає макси­мальне число символів, що може бути прочитане з вхідного потоку, доки не зустрінеться символ «  » (пропуск) або символ, який не може бути перетворений відповідно до заданого формату;

**f | n** — дозволяють приглушити погодження за замовчуванням про використану модель пам’яті («далека», «близька» пам’ять);

**h | l** — предикати, що визначають відповідно аргументи типів **short і long**;

**тип** — задається одним із символів: **d** — десяткове ціле; **і** — десяткове, вісімкове чи шістнадцяткове ціле зі знаком; **с** — одиночний символ;**u** — беззнакове десяткове число; **х, X** — беззнакове шістнадцяткове число; **0** — вісімкове число; **s** — сприймає символи без перетворення до символу **«\n»** або пропуску, доки не буде досягнута задана довжина (при виведенні видає до потоку всі символи до символу **«\0»** або до досягнення специфікованої точності); **f, F** — значення з плаваючою крапкою; **е, Е** — значення у експоненціальній формі; **G, g** — значення зі знаком у формі **f** або **е**.

Аргументи у функції **scanf** мають бути записані у формі покажчиків, тобто у вигляді **&х, &у, &mas[i]** тощо. Для введення змінних **k** (типу **int**) і **р** (типу **float**) цю функцію можна за­писати так:

**scanf(” %d  %f   \n ”, &k, &р);.**

Аргументами функції scanf є не імена змінних, а їхні адреси (символ & означає взяття адреси). Функція scanf зчитує коди з клавіатури, перетворює їх у відповідності із специфікацією форматного рядка і передає програмі. Форматний рядок і список аргументів для функції scanf – обов'язкові параметри. Описи відповідних функцій введення / виведення містяться у заготовочному файлы stdio.h.

Наприклад, програма обчислення площі кола:

#include <stdio.h>

void main()

{ double r,s;

const float pi=3.14;

printf("\n r=");

scanf("%lf",&r);

s=pi\*r\*r;

printf("\narea of a circle: %10.3f",s);

system("pause");

return 0; }

***Приклад 6.*** Ввести два числа та обчислити їх суму.

**#include <stdio.h>**

**#include <conio.h>**

**main( )**

**{ int a, b, c;**

**scanf ( " %d  %d",&a,&b);**     *//введення чисел 5 и 8*

**с = a + b;**

printf ("Summa =**%d \n", c);**

**getch();**      *//затримка екрана*

**}**

У результаті виконання програми буде виведено: **Summa=13**.

Форматний рядок наказує функції **scanf** ввести десяткове число, яке треба помістити в змінну **а**, а потім перейти до наступного не порожнього символу і з цього моменту почати введення нового десяткового числа, яке потім присвоюється змінній **b**. Якщо за рядком керування форматом аргументів більше, ніж специфікацій формату, зайві аргументи ігноруються. Коли для специфікацій формату недостатньо аргументів, результат не визначений.

У наведеному фрагменті програми для форматованого виведення даних використовується функція **printf**.

**Правила використання виразів**:

1. Вираз записується в один рядок.
2. У виразах використовуються тільки круглі дужки, причому кіль­кість дужок, що відкриваються, повинна відповідати кількості дужок, що закриваються.
3. Не можна записувати підряд два знаки арифметичних операцій.
4. Операції одного старшинства виконуються послідовно зліва направо.

Виведення значень виразів ми можемо здійснювати за допомогою операторів потокового введення-виведення **cin** та **cout** або форматованого введення-виведення **scanf** та **printf**.

В C++ оператори об’єднуються в блоки — функції. **Функція** — це послідовність операторів. Кожна програма в C++ повинна містити **головну функцію** main(). Саме з першого оператора в main() і починається виконання програми. Функції, як правило, виконують конкретне завдання. Наприклад, функція max() може містити оператори, які визначають максимальне число з двох переданих їй. Функція calculateGrade() може вираховувати оцінку студента..

**Порада**: Пишіть функцію main() в файлі .cpp з тим же іменем, що має проект. Наприклад, якщо ви пишете програму Chess, то помістіть вашу функцію main() в chess.cpp.

У загальному випадку програма складається з декількох функцій, що не перетинаються (тобто **«вкладення» однієї функції в іншу неприпустиме**). Перед функціями і між ними можуть бути присутні оголошення об’єктів даних і оператори препроцесорної обробки. Функції користувача, які викликаються у головній функції main(), слід обов’язково описати до їх використання. Наведемо приклад запису фрагмента простої програми:

//..............фрагмент простої програми на С++

#include <iostream.h>

#include <соnio.h> //директиви препроцесора

const int n=20;

void main()

{

float mas[n]; //опис одновимірного масиву

//..............введення елементів масиву

for (int i=0; i < n; i++)

cin >> mas[i];

//......виконання перетворень

//......виведення перетвореного масиву

for(int i=0; i < n; i++)

cout << " " << mas[i] << " ";

getch (); // затримка результату на екрані

}

**Коментарі** необхідні для пояснень призначення тих чи інших частин програми і їх текст завжди ігнорується компілятором. Мова С++ використовує два різновиди коментарів:

* *// текст* — **однорядковий коментар**, який починається з двох символів «/» («коса риска») і закінчується символом переходу на новий рядок;
* */\* текст \*/* — **багаторядковий коментар**, що розташовується між символами-дужками «/\*» і «\*/».

Багаторядкові коментарі не можуть бути вкладеними один в одний, а однорядкові коментарі можна вкладати в багаторядкові коментарі. Багаторядкові коментарі доцільно застосовувати для тимчасового виключення блоків при налагодженні програми.

Наведемо кілька порад стосовно раціонального складання коментарів:

* коментарі повинні бути добре складеними реченнями, мати правильну пунктуацію та містити тільки потрібну для супроводу інформацію;
* пропуск — один з найбільш ефективних коментарів, що значно поліпшує розуміння програми;
* штрихові лінії коментаря або порожні рядки застосовуються для поділу функцій та інших логічно завершених фрагментів програм.

Директива препроцесора **#include<iostream>** забезпечує підключення до програми засобів зв’язку зі стандартними потоками введення-виведення даних. Ці засоби знаходяться у заголовному файлі **iostream.h**, де **і (*input*)** — введення, **о (*output*)** виведення, **stream** — поток, **h (*head*)** — заголовок.

При створенні програми враховують такі основні вимоги:

* усі використані константи, змінні, функції та нестандартні типи повинні бути оголошеними (описаними) до їхнього першого використання, і ці оголошення можна розміщати в будь-якому місці програми;
* кожний оператор мови закінчується символом **«;»;**
* фігурні дужки (**« { »** та **« } »**) виділяють складений оператор і все, що подано між такими дужками, синтаксично сприймається як один оператор;
* вкладені блоки повинні мати відступ у 3-4 символи, при цьому блоки одного рівня вкладеності слід вирівняти за вертикаллю.

**Зовнішні оголошення**

*Зовнішні оголошення* – це глобальні описи даних (змінних або констант). Змінна у програмуванні є моделлю (зображенням) об’єкта в пам’яті комп’ютера. На фізичному рівні поняттю змінної відповідає група комірок оперативної пам'яті. Ці адреси ставляться у відповідність ідентифікаторам (іменам) змінних під час їх оголошення. Таким чином, ім'я змінної *вказує* (або *посилається*) на на першу комірку з групи, а величину групи визначає тип змінної (рис. 1).

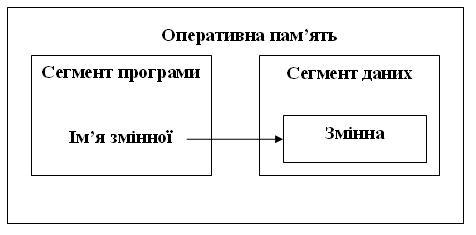
* 

Рис. 1. Ідентифікатор змінної та його асоціація з коміркою пам'яті

**Функція** – це частина програми, що виконує логічно завершений набір дій. В C/С++ вона є єдиним способом представлення підпрограм. Формат оголошення функції:

**тип\_результату ім'я([параметри]) *// заголовок функції***

**{ … } *// тіло функції***

Тіло функції може містити описи констант, змінних і оператори мови, які відокремлюються символом «;». Описи даних можуть зустрічатися в будь-якому місці функції, але до першого звертання до змінної. Якщо параметри у функції відсутні, то в заголовку все рівно записуються круглі дужки. Одна з функцій програми на мові C/С++ повинна мати ім'я main***.*** Саме з неї починається виконання програми. Функція main визначає дії, що виконуються програмою, і викликає інші функції. Якщо програма містить тільки одну функцію, то вона і є головною (має ім'я main).

Програма може містити довільне число директив препроцесора, оголошень і визначень змінних, описів функцій. Порядок появи цих елементів у програмі досить істотний: він впливає на можливість використання змінних, функцій і типів у різних частинах програми.

Логічні операції та операції порівняння в мові С/С++ оперують поняттями "істина" і "хиба"

**Областю дії оголошення змінної, тобто ідентифікатора змінної** є частина програми, в якій його можна використовувати для доступу до зв’язаної з ним області пам’яті. Залежно від області дії змінна може бути локальною або глобальною. Локальна змінна визначена всередині блока (блок розташований між фігурними дужками). Область її дії обмежена початком опису змінної та кінцем блока, включаючи усі вкладені блоки. Змінна, визначена поза будь-яким блоком, називається глобальною, і областю її дії вважається файл, у якому вона визначена від початку опису до його кінця.

***Для самостійного вивчення .*** Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

**Запитання та завдання**

**1.** Як забезпечується організація введення даних на С++?

**2.** Як здійснюється виведення результатів на С++?

**3.** Як реалізується потокове введення даних «**сіn>>**»? Наведiть приклади.

**4.** Охарактеризуйте основні аспекти використання потокового ви ведення даних «**соut<<**».

**5.** Що таке форматоване введення-виведення даних?

**6.** Поясніть правила застосування функції **scanf**.

**7.** Як здійснює роботу функція **printf**?

**Вправи для письмових відповідей (одна відповідь 0,25 балів)**

1. Що буде виведено на екран за такими інструкціями?

**cout<<9/5<<' '<<-9/5<<' '<<9/-5<<' '<<-9/-5<<endl;**

**cout<<9%5<<' '<<-9%5<<' '<<9%-5<<' '<<-9%-5<<endl;**

**cout<<9./5.<<endl;**

2. Що буде виведено на екран за такими інструкціями?

**cout<<7/3<<' '<<1/6<<endl;**

**cout<<7./3.<<' '<<1./6.<<endl;**

3. Припустимо, що значення дійсної змінної **length** відповідає довжині будівлі в міліметрах. Написати вираз, що задає довжину будівлі в метрах.

4. Нехай значення цілої змінної **sizeOfFile** задає розмір файлу в байтах. Написати вираз, значенням якого є розмір файлу в Кбайтах.

5. Нехай **v** – ім'я цілої змінної з невід'ємним значенням. Написати вираз, який обчислює: а) значення молодшої десяткової цифри числа **v**; б) значення молодшої двійкової цифри числа **v**.

6. Що буде виведено на екран за такими інструкціями?

**cout<<1+4/2<<' '<<(1+4)/2<<endl;**

**cout<<2\*4%7<<' '<<2\*(4%7)<<endl;**

**cout<<51/6%7<<' '<<51/(6%7)<<endl;**

**cout<<2\*(7%8)<<' '<<12/6%8<<' '<<5-3%2<<endl;**

7. Що буде виведено на екран за такими інструкціями?

**cout<<4\*6/8<<' '<<4/8\*6<<endl;**

8. Що буде виведено на екран за такими інструкціями?

**cout<<(-3+5)\*(2%7/3+4\*2)<<endl;**

9. Що буде виведено за такими інструкціями?

а) **int a=2, b=3;**

**cout<<"a\*b="<<a\*b<<";\n\*\*\*"<<endl;**

б) **int a=3, b; cout<<a\*a<<' '<<a+4<<'\n';**

10. Нехай **a**, **b**, **c** – дійсні змінні, що позначають довжини сторін трикутника. Запишіть вираз, що задає периметр, та інструкцію, яка присвоює периметр змінній **p**.

**11.** Яке із за коментованих значень не буде давати помилку компіляції?

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int main()**

**{ float fex;**

**fex=-123;**

**//fex=abs(fabs);**

**//fex=fabs(-123);**

**//fex=abs(-123);**

**cout << fex << endl;**

**return 0;}**

Результати надсилати на електронну адресу викладача

**t.i.lumpova@gmail.com** у вигляді файлу з іменем у форматі

**<Номер групи> <Номер Лекції ><Прізвище англійською>**

Наприклад, PTBD21-Лекція01Ivanov.docx.

Тему в заголовку листа записати

**AП<Номер групи>- Л<Номер лекції>-<Прізвище>**

**Строк відсилки відповідей 04.03.2024**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, тему в заголовку листа записати

**АП<Номер групи>-Запитання-<Прізвище>**.

***Література***

1. Вступ до програмування мовою С++. Організація обчислень: навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставовський. – К. : Видавничо-поліграфічний центр "Київський університет", 2012. – 175 с. с.: іл. ISBN (укр.) . URL: <http://csc.knu.ua/uk/library/books/belov-24.pdf>
2. Джейс Либерти Освой самостоятельно С++ за 21 день: 3-е изд. пер. с англ.: Уч. пос. – М.: Издательский дом „Вильямс”, 2001. – 816 с.: ил..
3. Бондарев В. М. Программирование на С++: Учеб. пособие. — Харьков: СМИТ, 2004г. — 294 с. URL: <https://www.rulit.me/author/bondarev-v-m/programmirovanie-na-c-get-161082.html>
4. Липпман С. Б., Лажойе Ж. Язык программирования С++: Вводный курс. — М.: ДМК, 2001. URL: <http://www.insycom.ru/html/metodmat/inf/Lipman.pdf>
5. Дейтел Х., Дейтел П. Основы программирования на С++. – М.: Бином, 1999. – 1024 с.