Лекція 1. Основи синтаксису та алгоритмічні конструкції C/C++

1. Структура простої програми на мові С++.

2. Потоковий ввід та вивід даних.

3. Оператор присвоєння. Використання змінних.

4. Алгоритмічні конструкції в С/С++.

1. Структура простої програми на мові С++

Продемонструємо приклад найпростішої програми на мові С++, яка запитує у користувача два цілочисельних значення для змінних a і b, аналізує їх і виводить більше число. В поданому нижче коді програми, у порівнянні з аналогічним кодом у вступі, дотримуємося стилю, характерного для С++.

```
#include <iostream>
using namespace std;
int main(void)
  {
    int a,b,max;
    cout << "a=";      //запрошення ввести значення a
    cin >> a;          //ввід значення змінної a
    cout << "b=";      // запрошення ввести значення b
    cin >> b;          // ввід значення змінної b
    if(a>b) max=a;       //якщо a>b то max=a
    else max=b;          //в протилежному випадку max=b
    cout << "max="<<max;  //вивід максимуму
    return 0; //при успішноу виконанні повертаємо 0
}
```

Перший рядок включає (***include*** - включити) до тексту програми так званий заголовковий (h від ***header*** - заголовок) файл. У цьому файлі описані бібліотечні функції та класи. Використовуючи ці описи, компілятор перевіряє правильність виклику системних функцій. У нашому прикладі програма

використовує системні об'єкти для введення (**cin**) і виводу (**cout**) даних, описи яких знаходяться в заголовному файлі **iostream.h**. Назви заголовкових файлів найчастіше утворюються від деяких абревіатур англійських слів, їх корисно навчитися розуміти, а не запам'ятовувати. У нашому прикладі: io - input / output (ввід / вивід), stream (потік), con (console - пульт оператора, тобто клавіатура і дисплей).

Другий рядок вказує, що бібліотечні функції містяться в просторі імен std (систему просторів імен запровадили в новіших стандартах С++).

Третій рядок містить заголовок функції **main**. Функція з такою назвою повинна бути в кожній програмі на мові С/С++. Саме з неї починається виконання програми. Службове слово int (від integer - цілий) повідомляє, що результатом роботи функції main повинно бути ціле число, за яким операційна система, запустивши програму **main**, може "вирішити", правильно чи неправильно завершилася робота програми. За загальноприйнятою угодою нульове значення, що повертається функцією main, свідчить про нормальне завершення роботи програми. Службове слово **void** (дослівно - порожнеча), зазначене в круглих дужках, повідомляє, що у функції main аргументи відсутні, в нових стандартах є необов'язковим, тобто після назви функції дужки можуть бути порожніми.

Текст програми (тіло функції) розміщується в фігурних дужках (4-та 14-та стрічки). В п'ятій стрічці оголошено три змінні з іменами **a**, **b** та **max**, які можуть набувати цілочисельних значень. Окрім типу int в С/С++ можна використовувати і інші цілі типи даних, що відрізнятимуться діапазонами допустимих значень. Для десяткових дробів у найпростішому випадку використовуватимемо тип **float** (коротке дійсне, 4 байти) або **double**.

Шоста стрічка є першою стрічкою програми, яка виконує якусь дію – вона виводить на дисплей повідомлення, що складається з двох символів (a =). Текст повідомлення обмежують подвійними лапками. Рядок 7 організовує зупинку роботи програми до тих пір, поки користувач не набере на клавіатурі якесь число і натисне клавішу Enter. Отримане значення буде прийняте лише, якщо воно є цілим числом, і програма його розмістить в змінній a. У схожий спосіб в рядках 8 і 9 буде організовано ввід значення числової змінної b.

Використаний в цьому прикладі спосіб вводу та виводу належить виключно до С++, – об'єкти **cin** та **cout** є екземплярами відповідних класів, в мові С, як відомо, класи та об'єктно-орієнтована парадигма не підтримується. Такий спосіб є дещо простішим, за використання функцій **printf** та **scanf_s**, тому в подальшому ми надаватимемо перевагу саме йому.

Далі організовано порівняння поточних значень змінних **a** і **b**. Якщо значення змінної **a** більше, то воно присвоюється змінній **max**, в іншому випадку в змінну **max** заноситься значення **b**.

Рядок 12 виводить на дисплей два повідомлення - текстове "max =" і числове (значення змінної max).

Часто, щоб затримати на екрані повідомлення програми до тих пір, поки користувач не натисне будь-яку клавішу програмісти вдаються до виклику функції getch (getch - від get character, дай символ) перед оператором завершення функції. При використанні MS Visual Studio потреба в такому "трюкові" відпадає, оскільки тут є команда "Run without debugging" (Ctrl+F5), виконання якої зупиняє консольне вікно доки користувач не натисне будь-яку клавішу.

Останній рядок повертає керування операційній системі і видає в якості значення функції число 0.

Загальні правила написання програми на C/C++:

- якщо програма використовує якісь бібліотечні функції, то в перших її рядках повинна бути вказівка підключити відповідні заголовкові файли;
- програма може містити більш ніж одну функцію, але серед них обов'язково має бути функція з іменем ***main***;
- кожен рядок програми, що містить оголошення чи виконувану дію, закінчується крапкою з комою;
- тіло функції утворюється з команд, які виконуються під час її виклику і поміщається у фігурні дужки.

2. Потоковий ввід та вивід даних

Для програмістів-початківців найпростіше організувати в програмі введення числової інформації з буфера потоку, пов'язаного з так званим стандартним пристроєм введення (***stdin***). Дані в цьому буфері з'являються в той момент, коли програма звертається до користувача і очікує закінчення набору затребуваних числових значень з клавіатури. Програма може запитати одне або кілька значень:

```
cin >> d;
.....
  cin >> x1 >> x2 >> x3;
```

Перший рядок виконує запит вводу значення однієї змінної d. Наступний рядок програми є запитом на ввід одразу трьох числових значень, перше з яких буде присвоєно змінній x1, друге значення призначається для змінної x2, третє значення - для змінної x3. У відповідь на запит програми користувач повинен набрати на клавіатурі три числа, розділяючи їх, принаймні, одним пропуском. Набір рядка з числами потрібно завершити натисканням клавіші Enter. Кількість числових значень, які набираються користувачем в межах рядка, може виявитися як меншою, так і більшою, ніж потрібно програмі. У першому випадку продовження програми затримається до тих пір, поки користувач не введе достатню кількість даних в наступних рядках. Якщо користувач набере занадто багато значень у введеному рядку, то зайві числові дані збережуться у

буфері вводу і будуть передані програмі при виконанні наступного оператора cin. Тип числового значення, що вводиться і його величина мають відповідати типу змінної, в яку це значення має бути введене. Турбота про це цілком покладається на користувача, тому правилом хорошого тону для програміста є вивід детальних пояснень перед кожним вводом даних з клавіатури.

Потоковий ввід даних є альтернативою використання функції С ***scanf*** і належить до засобів С++. Для забезпечення потокового введення до програми слід підключити заголовний файл iostream.h:

```
#include <iostream>

using namespace std;

  int main()

{

  int i;

  float f;

  double d;

..........

  cin >> i >> f >> d;

}
```

Для початківців оптимальною є організація потокового виводу числової інформації на стандартний пристрій виводу (***stdout***) організований в С++ програмно у вигляді об'єкта ***cout*** класу ***ostream***:

```
#include <iostream.h>

using namespace std;

int main()

{ int i;

  float f;

  double d;

..........

  cout << i+1 << f << f*d;

..........

  return 0;

}
```

Тут не треба піклуватися про формати виводу результатів. Для кожного типу даних існують відповідні системні угоди. Перехід в початок наступного

рядка тут здійснюється шляхом включення в список виведення або згаданого вище керуючого символу ' \n ', або аналогічного йому символу кінця стрічки endl:

```
cout << i+1 << f << f*d << endl;
```

Виведення даних у списку виводу може супроводжуватися пояснювальними підписами:

```
cout << "x1="<< x1 << "x2=" << x2 << '\n';
```

3. Стандартний вигляд оператор присвоєння, використання змінних

Оператор присвоєння є найбільш поширеним засобом, що дозволяє змінити значення змінної під час роботи програми. У найпростішому випадку він має формат:

namev = expression;

Тут ***namev*** – ім'я змінної, ***expression*** – вираз, значення якого буде присвоєно змінній ***namev***. Якщо змінна ***namev*** відноситься до категорії числових даних, то результат обчислення виразу теж повинен бути числовим. Типи змінної ***namev*** і тип значення виразу при цьому можуть не співпадати. Про відповідне перетворенні машинних форматів система подбає самостійно, проте, якщо, наприклад, виконується перетворення дійсного значення в цілочисельне, дробова частина виразу (якою б вона не була) буде відкинута. У тих випадках, коли тип виразу допускає більш широкий діапазон представлення даних, можлива втрата точності або переповнювання діапазону, передбаченого для змінної ***namev***. У таких випадках компілятор видає повідомлення про можливі наслідки, і на такі повідомлення програміст зобов'язаний звернути увагу.

Якщо декільком змінним необхідно присвоїти значення одного і того ж виразу, то оператор присвоєння допускає розширений формат:

v1=v2=v3=exp;

Використовуючи таку форму присвоєння, не варта писати оператори, результат виконання яких може тлумачитися по різному, наприклад:

```
i=5;

i=a[i]=4;

j=6;

a[j]=j=2;
```

Результат таких дій залежить від алгоритму перегляду стрічки програми і послідовності виконання присвоювання – зліва направо, чи справа наліво. Слід уникати подібних ситуацій, тому, що у випадку перенесення програми в іншу систему програмування можна отримати несподіваний результат.

Змінні, що використовуються в програмі, обов'язково повинні бути оголошені до їх використання в тих чи інших виконуваних операторах. На відміну від мови Pascal алгоритмічні мови С, С + + дозволяють вводити такі

описи не тільки на початку програмних одиниць (функцій), але і по ходу формування програми:

```
int main(void)
{
  int i,j;
..........
  float s=0;
  for(int k=0; k<10; k++)
  {
    ......................
  }
  ......................
}
```

У наведеному прикладі змінні ***i*** та ***j*** оголошені на початку функції, а оголошення змінної ***k*** зустрілося в операторі циклу. За стандартами C/C++ місце оголошення змінної визначає область її актуальності. Змінні, описані на початку функції (такі як ***i*** та ***j*** в наведеному вище прикладі), вважаються локалізованими в даній функції і можуть бути використані в будь-якій частині тіла цієї функції. Повторне оголошення змінних з такими ж іменами вважається помилкою (дублювання імен змінних). На відміну від цього дія змінних, оголошених в деякому внутрішньому блоці функції (у нашому прикладі змінна ***k*** оголошені в циклі ***for***), поширюється тільки на час роботи цього блоку. Тобто після виходу з циклу пам'ять, виділена під змінну ***k***, повертається системі, і змінною k, без повторного оголошення, користуватися вже не можна.

Оголошення змінної можна поєднати із присвоєнням їй початкового значення (ініціалізацією):

```
int x = 18, y = -5;
float a = 5f;
```

Для оголошення іменованих констант зазвичай використовують наступну конструкцію:

const [tc] namec = value;

тут

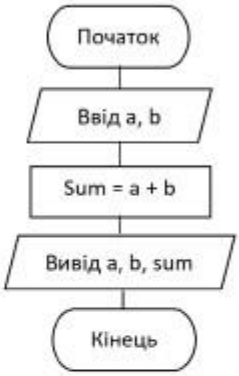
- ***tc*** - необов'язковий тип константи (за замовчуванням tc = int);
- ***namec*** - ім'я константи;
- ***value*** - значення константи.

Іноді для задання константних даних вдаються до механізму найпростішої макропідстановки:

```
#define Nmax 100
#define eps 1e-6
```

Це означає, що перед трансляцією програми компілятор (точніше, прекомпілятор) прогляне її текст і всюди, де зустріне поєднання символів Nmax, замінить його на число 100, а поєднання символів eps на число 1e-6. Результат буде тим же самим, але робота з макропідстановкою пов'язана з більш помітними витратами часу під час компіляції коду. Хоча при такому способі програма не задіює механізм зберігання даних в пам'яті, на відміну від використання іменованих констант.

4. Алгоритмічні конструкції в C/C++.



Загальновідомо, що для побудови алгоритмів розв'язування задач та їх програмування використовуються три базові алгоритмічні конструкції: ***лінійна***, ***розгалужена*** та ***циклічна***. Кожна мова програмування має відповідні інструменти для реалізації кожної з них.

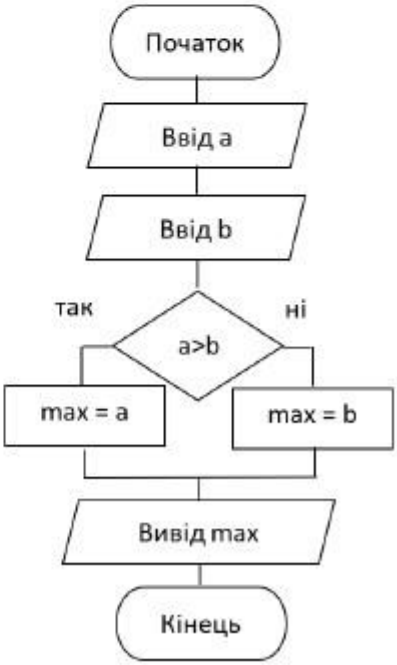
Лінійна алгоритмічна конструкція полягає у виконанні команд послідовно (в лінію) одна за одною. До прикладу, таку структуру має функція main з прикладу у вступній частині (див. блок-схему на малюнку).

Розгалужена алгоритмічна конструкція передбачає виконання програми за однією з кількох альтернативних послідовностей команд залежно від виконання деяких умов. Розгалуження організовують за допомогою умовного оператора, в C/C++ його синтаксис такий:

```
if(умова) {S1; S2;...}
[else {Q1;Q2;...}]
```

Якщо умова, зазначена в круглих дужках, справджується, то виконується послідовність операторів S1, S2, ..., в протилежному випадку виконується послідовність – Q1, Q2, ... Послідовності операторів і після if і після else поміщаються у фігурні дужки. Якщо така послідовність містить лише один оператор, то дужки необов'язкові:

```
if(умова) ОператорS;
[else ОператорQ1;]
```



Альтернативної вітки разом із оператором **`else`** може не бути. Тоді мова йде про виконання або обхід послідовності операторів S1, S2, ... Такий варіант розгалуження називають неповним, або ***короткою формою розгалуження***. Конструкція з оператором **`else`** називається ***повною формою розгалуження***, відповідно. В прикладі на початку цієї лекції повна форма розгалуження використана для знаходження більшого з чисел **`a`** та **`b`**. Блок-схема алгоритму роботи цієї програми зображена на малюнку.

Прості умови задаються за допомогою операцій порівняння:

```
if(a>b) cout<<a;

else cout<<b;

if(x>=0) y=sqrt(x);
```

Для порівняння двох значень на рівність використовують знак "**==**", а відношення "не дорівнює" задається знаком "!=".

Більш складні умови можуть складатися з елементарних співвідношень за допомогою логічних операцій "АБО" (**||**), "І" (**&&**), "НЕ" (**!**). Наприклад, перевірка належності x діапазону [a, b] виглядає наступним чином:

```
if(a<=x && x<=b) {...}
```

У мові С логічні значення відсутні, натомість діє угода про те, що істині відповідає будь-яке ненульове числове значення, а хибність, відповідно, еквівалентна нулю. Тому в якості умови іноді можна зустріти звичайний арифметичний вираз:

```
if(a){...}
```

Така перевірка еквівалентна співвідношенню:

```
if(a != 0) {...}
```

Оператори циклу дозволяють організувати повторне виконання фрагмента програми до тих пір, поки не виконається деяка умова. Найпростіша конструкція оператора циклу в C/C++ починається зі службового слова **`while`** (від англ. – поки):


```
while(умова) {Q1; Q2;...}
```

Вхід в такий цикл починається з перевірки умови, записаної в круглих дужках. Якщо ця умова справджується, то тіло циклу (оператори Q1, Q2, ...) виконується. У разі невиконання умови тіло циклу не виконується жодного разу. Як і у синтаксисі оператора **`if`**, тіло циклу може складатися тільки з одного оператора, тоді фігурні дужки є необов'язковими.

В наступному фрагменті коду показано обчислення суми десяти елементів **`a[i]`** (i = 0,1,…,9) масиву цілих чисел **`a`** за допомогою оператора **`while`**:

```
int s=0, i=0;
while(i<10)
{
s=s+a[i];
i++;
}
```

Використання оператора циклу ***while*** в задачах, аналогічних до розглянутої вище, потребує додаткових дій в програмі. Перед циклом – задання початкових значень суми ***s*** та номера елемента ***i***, а в тілі циклу збільшення номера елемента ***i*** на 1 оператором ***інкременту i++*** (виконує дію ***i=i+1***). Для таких ситуацій в мові С/С++ передбачено більш універсальну конструкцію циклу ***for***:

```
for(S1; C; S2) { Q1; Q2;...}
```

тут

- S1 – дії, які виконуються перед першим виконанням групи операторів Q1, Q2, ..., утворюють тіло циклу;
- C – умова наступного повторення тіла циклу. Якщо умова С не виконується, то тіло циклу оминається і цикл завершується;
- S2 – дії, що виконуються після останнього оператора тіла циклу. Після цього керування передається на перевірку умови повторення циклу С.

Оператор ***for*** дозволяє записати попередній приклад з обчисленням суми елементів масиву набагато компактніше:

```
int s=0;
for(int i=0; i<10; i++) s=s+a[i];
```

У наведеному коді тіло циклу складається з єдиного оператора, який можна не брати в фігурні дужки. Перед початком виконання циклу виконується початкове присвоєння змінній ***i*** значення 0. Після кожної ***ітерації***, тобто, виконання тіла циклу значення ***i*** збільшується на 1. Ще одна особливість цієї реалізації, – змінна ***i*** оголошена в циклі і не існує поза його межами, це дозволяє, наприклад, оголошувати і використовувати ще одну змінну ***i*** в наступному циклі. Блок-схемою, наведеною для зображення циклічної алгоритмічної конструкції з оператором ***while*** можна так само зображати і циклічну конструкцію ***for*** в загальному випадку.

Форма оператора ***for***, використана в прикладі дозволяє організувати повторення тіла циклу задану кількість разів, змінюючи при кожному повторенні значення деякої керуючої змінної. Іноді таку змінну називають лічильником циклу або ***параметром циклу***, а сам цикл, – ***циклом з параметром***. В деяких алгоритмічних мовах програмування (наприклад Basic, Pascal) цикл з ключовим словом ***for*** реалізовані виключно для

програмування таких циклів, в C/C++ конструкція **`for`** є універсальною і дозволяє реалізовувати цикли будь-якої форми, так само, як оператор **`while`**.

Третя конструкція циклу починається з оператора do і завершується перевіркою умови продовження циклу while:

```
Do
{
 Q1;
 Q2;
 ...
} while(умова);
```

Відмінність конструкції do - while від двох попередніх операторів циклу полягає в тому, що тут тіло циклу виконується, принаймні, один раз. Цей цикл іноді супроводжують терміном цикл з післяумовою – тобто перевірка умови продовження циклу проводиться після виконання його тіла.