## ЕЛЕМЕНТИ МОВИ С++

### 2.1. З історії створення мови С++

Історія мови С++ (Сі-плюс-плюс) почалася з мови С (Сі). Компанія Bell Laboratories на початку 1970-х рр. створила мову С як інструмент розробки операційної системи UNIX. Спочатку Мартін Річардс розробив мову BCPL, потім на її основі Кен Томпсон – мову B, і нарешті, Денніс Річі – мову С. Завдяки лаконічності, виразній потужності й надійним компіляторам ця мова програмування дуже швидко стала однією з найпопулярніших і найпоширеніших. Проте з часом розміри й вимоги до програм зростали, тому можливості мови С та інших мов, подібних до неї, поступово вичерпувалися.

Кризу в програмуванні, що намітилася в кінці 1970-х рр., було подолано за допомогою, головним чином, *об'єктно-орієнтованого програмування*. Однією з об'єктноорієнтованих мов була мова С++, яку на базі мови С розробив Бйорн Страуструп у компанії Bell Laboratories у кінці 1970-х рр.

У наступні два десятиліття мову С++ було збагачено й стандартизовано. Її втілено в кількох сучасних системах програмування, зокрема в Microsoft Visual C++ (існує кілька версій цього дуже потужного засобу програмування), і на неї орієнтуються автори в прикладах цієї книги.[[1]](#footnote-1)

Нащадками мови С++ є мови Java (Джава) та C# (Сі-шарп або Сі-діез), спеціалізовані для програмування в сучасних комп'ютерних мережах. Ці мови за структурою схожі на С++, тому, володіючи С++, неважко перейти на Java або C#, і навпаки.

### 2.2. Перша програма – "Hello, World!"

Програма мовою С++ записується у файл з розширенням

.**cpp,** наприклад **prog001.cpp**.

**Приклад**. У книжках із мов програмування дуже часто першою наводять програму, яка "вітає світ", тобто виводить на екран рядок-привітання.

**#include <iostream> using namespace std; int main () {**

**cout<<"Hello, World!"; return 0;**

**}**

**prog001.cpp**

У першому рядку записано **директиву препроцесора**. **Препроцесор** – це складова частина компілятора, яка проводить попередню обробку програми. Директиви записують в окремих рядках і починають символом #. Слово **include** (включити) означає, що препроцесор перед компіляцією програми має включити в неї вміст спеціального файлу зі складу системи програмування, ім'я якого **iostream** записане в кутових дужках. У цьому файлі оголошено засоби введення й виведення (ім'я **cout**, операцію **<<** і багато інших). Без включення цього файлу ім'я **cout** буде невизначеним, і компілятор повідомить про цю помилку.

Файл **iostream** є одним із багатьох **заголовних** (*header*) **файлів** (або **h**-файлів), що входять до складу системи програмування, тобто є стандартними. У директивах імена стандартних заголовних файлів записуються в кутових дужках. Багато стандартних заголовних файлів має порожнє розширення, для решти традиційно використовують розширення **h**.

У другому рядку розташовано інструкцію компілятору "використати простір імен **std**". Не пояснюючи значення слів "простір імен" (детальніше див. підрозд. 9.3), скажемо лише, що простір імен **std** є стандартним. У сучасних системах програмування мовою С++ у ньому описано всі бібліотечні засоби останнього покоління. Завдяки наведеній інструкції спрощується доступ до бібліотечних засобів (один з них, з ім'ям **cout**, використовується в програмі). Проте компілятори попереднього покоління цієї інструкції "не розуміють", тому для них писати її не можна.

Наведена програма складається з однієї функції з ім'ям **main**. Слова **int main()** у третьому рядку – це **заголовок функції**. Дужки **()** після імені **main** указують, що це ім'я саме функції. Ім'я **int** перед іменем функції є скороченням слова integer і означає, що функція має повертати ціле значення.

Функція з ім'ям **main** називається **головною функцією**. Вона має бути в кожній програмі, адже саме з неї починається виконання програми й зазвичай нею закінчується. Ім'я **main** не є зарезервованим, але використовувати його з іншим призначенням не слід.

Вміст рядків 4–7 утворює **тіло** функції, що починається символом **{** і закінчується **}**. У тілі функції задано дії у вигляді **послідовності інструкцій**. Інструкція в п'ятому рядку задає виведення на екран повідомлення **Hello, World!**. Воно з'являється у вікні програми, яке має відкритися на екрані під час її виконання та зникнути після завершення. Текстове повідомлення, що виводиться на екран, записується в лапках "".

За інструкцією в шостому рядку функція має повернути значення 0 (нуль)[[2]](#footnote-2).

Отже, запустимо програму на виконання. Чи встигнемо ми щось побачити? Відповідь залежить від системи програмування, що використовується. Деякі, але не всі, системи програмування дозволяють переглядати вікно програми після її завершення. З іншого боку, виконання програми можна затримати за допомогою бібліотечної функції **system.**

**#include <iostream> using namespace std; int main () { cout<<"Hello, World!"; cout<<endl; system("pause"); return 0;**

**}**

**prog002.cpp**

За інструкцією **system("pause");** виконання програми призупиняється й на екрані з'являється повідомлення, що треба натиснути будь-яку клавішу. Після натискання програма завершується. Завдяки попередній інструкції **cout<<endl;** повідомлення виводиться в новому рядку. Якби цієї інструкції не було, повідомлення з'являлося б відразу після слів **Hello, World!.** 3

Наведена програма складається з двох директив і головної функції. Узагалі програма може містити багато функцій, директив і деяких інших елементів.

**Вправа 2.1**. З'ясуйте, що буде виведено за нижченаведеними програмами. Поясніть різницю між ними.

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

**int main() { int main() { int main() {**

**cout<<"Microsoft"; cout<<"Microsoft "; cout<<"Microsoft"; cout<<"Visual"; cout<<"Visual "; cout<<endl; cout<<"Studio"; cout<<"Studio"; cout<<"Visual"; cout<<endl; cout<<endl; cout<<endl;**

**system("pause"); system("pause"); cout<<"Studio";**

**return 0; return 0; cout<<endl;**

**} } system("pause");**

**return 0;**

**}**

### 2.3. Алфавіт і словник мови С++

Кожна мова має **алфавіт** – скінченну множину символів, дозволених для використання. Алфавіт для написання програм мовою C++ містить:

* **літери** (**букви**) – великі й малі латинські **A, B, … Z, a, b, … z,** а також символ підкреслення **\_** та "долар" **$**;
* **десяткові цифри** **0, 1, 2, …, 9;**
* **знаки пунктуації** **+ - \* ~ / \ % ? ! = < > & | ^ # .**

**, ; : ' " ( ) [ ] { }** ; – деякі інші символи.

Із символів алфавіту утворюються **лексеми** – "слова", тобто послідовності символів, що розглядаються як неподільні й мають власний зміст. Серед лексем мови С++ виділяють константи, імена, знаки операцій тощо.

**Константа** (*constant*) – це позначення значення (числового або іншого). Константи також називають **літералами** (*literals*).

Нам добре відомі числові константи, наприклад **273, 3.1415926, 2.71828** (ціла частина від дробової відокремлюється крапкою, а не комою). В апострофах записуються символьні константи, у лапках – рядкові, наприклад **'3'**, **'A'**, **'\n'**, **"University"**, **"3.1415926"**, **"".** Докладніше константи описано нижче.

**Ім'я** – це послідовність літер алфавіту й цифр, що починається з літери[[3]](#footnote-3), наприклад **A, x1, \_1a$2, best\_student.**

Великі й малі літери в іменах *відрізняються*: **Name,** **NAME, name** – це різні імена. Довжину імен *не обмежено*[[4]](#footnote-4).

Ім'я завжди *іменує* якийсь об'єкт, видялючи з-поміж інших, тобто *ідентифікує* його, тому ім'я ще називають **ідентифіка-**

**тором**.

Деякі імена (англійські слова або їх скорочення) мають спеціальний зміст, наприклад **const, else, int,** **sizeof.** Вони називаються **зарезервованими словами**. Використовувати їх за іншим призначенням не можна (цього просто не дозволить компілятор). Зарезервовані слова інколи вважають окремим різновидом лексем.

**Знак операції**, або **оператор** (*operator*) – це позначення операції, виконання якої над числами та іншими значеннями породжує число або інше значення. Значення, до яких застосовується операція, називаються її **операндами** (*operand*), а породжуване значення – **результатом** (*result*).

Операції позначають окремими символами, наприклад **+** або **–**, а також іменами або послідовностями інших символів, наприклад **sizeof** або **<=**.

Лексеми часто відокремлюються так званими **порожніми символами**, які не мають зображення. Вони з'являються в тексті програми, якщо, набираючи його, натискати клавіші **Space** (пробіл), **Enter** (кінець рядка) і **Tab** (символ табуляції). Будь-які дві лексеми можна відокремити порожніми символами в довільній кількості. Компілятор, обробляючи текст програми з деякої поточної позиції, завжди виділяє найдовшу можливу лексему. Тому порожні символи між сусідніми лексемами обов'язкові лише тоді, коли, записані разом, ці лексеми утворюють лексему. Наприклад, записи **1+2, sizeof–2** та **1<=2** позначають застосування відповідних операцій і не є лексемами, а **sizeof2** є ім'ям, тобто **однією** лексемою.

1. Є також інші системи програмування на базі C++, наприклад Dev-C++ або Code::Blocks, які мають відмінності в інтерфейсі й синтаксичних деталях.

   Зауважимо: Microsoft Visual C++ має найкращу довідкову систему. [↑](#footnote-ref-1)
2. Повернути можна також інше ціле значення. Значення, що повертається функцією **main**, передається операційній системі як результат виконаної програми. За традицією 0 є ознакою вдалого виконання програми. [↑](#footnote-ref-2)
3. Символ підкреслення \_ може бути першою літерою імені, але імена, що починаються з одного чи двох \_, мають у C++ спеціальне призначення. Тому використовувати імена з \_ на початку не рекомендується. [↑](#footnote-ref-3)
4. Формально довжину імен у C++ не обмежено, але, наприклад, компілятор Microsoft Visual C++ 2005 розрізняє лише перші 2048 символів імені. [↑](#footnote-ref-4)