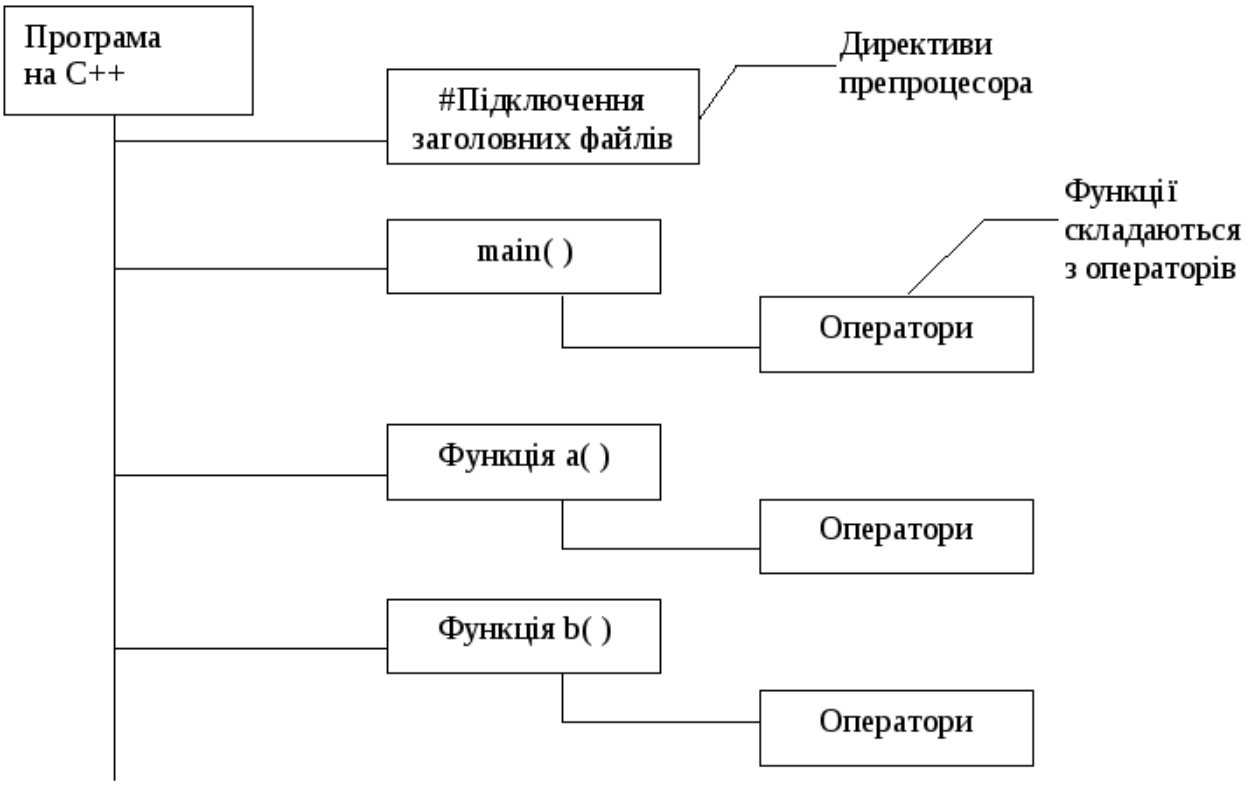
Лекція 5 - 6

Структура програми на мові С

Загальну ***структуру програми*** на мові С, С++ можна представити наступним чином:

1. препроцесорні команди;
2. опис глобальних змінних (необов'язково). Глобальні змінні – це ті змінні, які доступні (видимі) будь-якій програмній одиниці;
3. тіло програми



Основною структурною одиницею програми на мові С, що складає тіло програми є ***функція*** **–** це самостійна одиниця програми, яка спроектована для реалізації конкретної підзадачі. Функція є підпрограмою, яка може міститися в основній програмі, а може бути створена окремо (в бібліотеці). Кожна функція виконує в програмі певні дії. Всі функції можна розбити на:

1) стандартні функції, що входять до складу мови;
2) функції користувача (підпрограми, які виконують певні дії і складаються користувачем за необхідність);
3) головна функція, яка є точкою входу в програму і є обов'язковою складовою будь якої програми.

Кожна ***функція має наступну структуру***:

1) заголовок;
2) опис зовнішніх змінних (необов'язково)
3) тіло функції.

```
ТипЗначенняЩоПовертається Ім’я (СписокФормальнихПараметрів)
{ Опис локальних змінних
      Тіло функції
          return(ЗначенняЩоПовертається); }
```

Заголовок функції має наступну структуру:
тип ім’я(список параметрів),
- тип – це тип значення, який функція повертає;
- ім’я – це ідентифікатор за яким до функції звертаються;
- список параметрів – перелік (через кому «,») змінних, які передаються у функцію. Список параметрів є не обов’язковим, тоді дужки є пустими.

Тіло функція представляє собою сукупність операторів та команд, які беруться у фігурні дужки (представляють собою складений оператор).

Завершується функція оператором ***return***, який повертає результат виконання функції.

Головна функція завжди має фіксоване ім’я – ***main()***.

Кожна змінна або функція (кожне ім’я), що з’являється в програмі повинно бути описане. Тому часто на початку описують всі функції, а в кінці ставлять головну функцію. Частіше, головну функцію ставлять на початку програми, а за нею описують функції (вважається, що так зручніше), але тоді перед головною функцією необхідно описати прототипи функцій.

Питання, стосовно створення функцій та роботи з ними будемо розглядати пізніше.

Структура програм дещо відрізняється в залежності від середовища програмування. Ми орієнтуємося на IDE Microsoft Visual Studio, і з цього приклади програм будуть показані саме для MVS.

Приклад структури програми MVS з підключеними бібліотеками.

```
#include "stdafx.h"
#include <iostream> // підключення бібліотеки вводу-виводу
using namespace std; // доступ до простору імен стандартних бібліотек
int main()
{
}
```

Ім'я бібліотек, що підключаються, пишеться всередині знаків більше, менше. Заголовки і ім'я підключаються бібліотек – синоніми.

Синтаксис підключення заголовних файлів: #include <ім'я заголовку>

Більш старі заголовки підключаються так (цей стиль підключення бібліотек успадкований у мови програмування С):

#include <ім'я заголовку файла.h>

Різниця полягає в тому, що після імені ставиться розширення .h.

Наступна директива ***using*** відкриває доступ до простору імен (англ. Namespace) std, в якому визначаються засоби стандартної бібліотеки мови С ++. (Простір імен – деяка множина, створена для логічного угруповання унікальних ідентифікаторів).

За відправну точку виконання будь-С ++ - програми є функція main(). Функція містить чотири елементи:

- повертається тип (в нашому випадку int);
- ім'я функції (main);
- список параметрів, взятий у круглі дужки (в даному випадку список порожній);
- укладену в фігурні дужки, тіло функції, що представляє собою блок інструкцій. Інструкцією називається частина програми, яка визначає дію і не є директивою препроцесора.

Звернемо увагу на те, що кожна інструкція в мові С ++ закінчується крапкою з комою. Існують такі винятки:

- директиви препроцесора, що починаються з символу #;
- складені оператори, які обрамлені фігурними дужками.
- заголовок функції.

Для пояснення призначення структурних одиниць програми застосовуються коментарі.

Коментар це зрозуміла для програміста анотація в коді комп'ютерної програми. Існує дві нотації опису коментарів:

1) /* початок коментарю коментар може бути записаний
в декількох рядках*/ кінець коментарю
2) // коментар записаний в одному рядку.

Порядок виконання програми на мові С, С++

Програма, яка вже готова до виконання, зберігається уфайлі з розширенням **.exe**. Програма, від початкового коду, написаного на мові С до виконання проходить певний шлях перетворень, які представлені на схемі нижче.

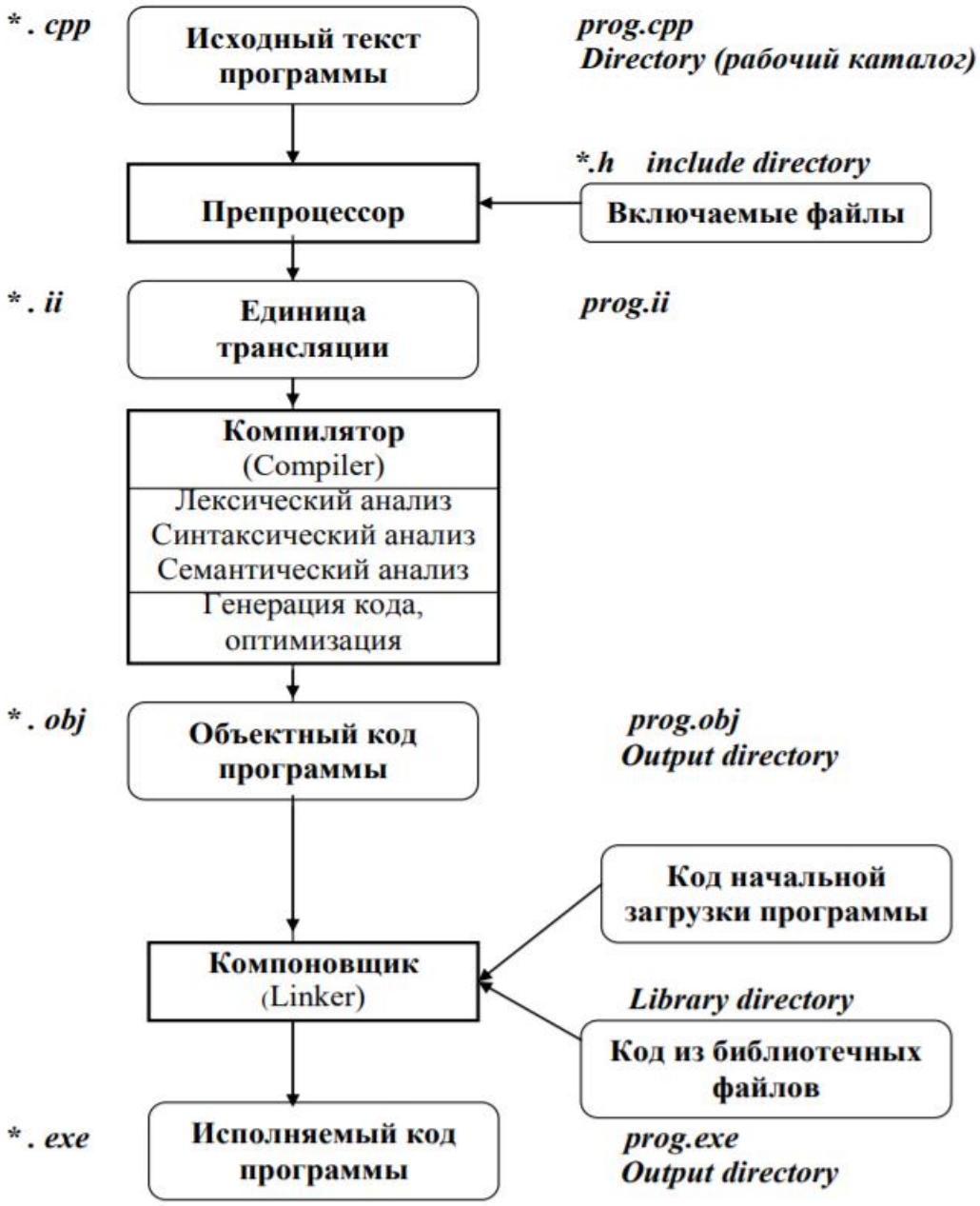
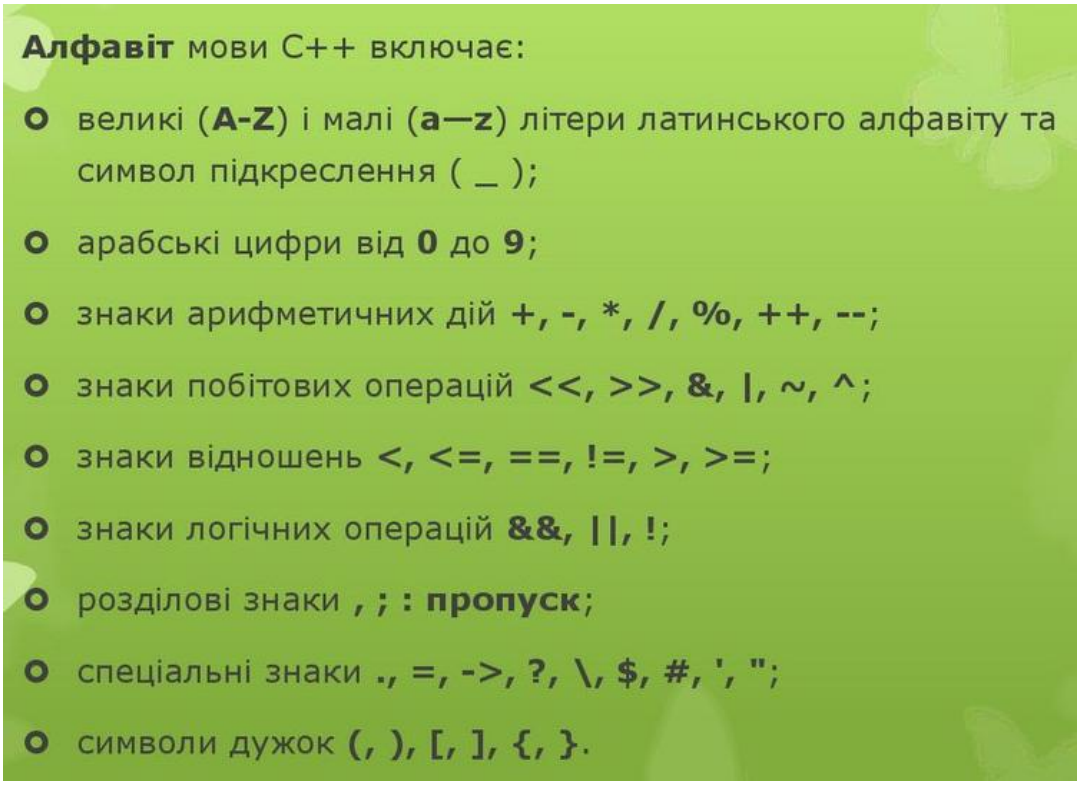


Рис. Схема підготовки програми до виконання.

Елементи мови програмування С, С++

Під елементами мови розуміють його базові конструкції, які використовуються при написанні програм. До них відносяться: алфавіт, правила запису констант і ідентифікаторів, основні типи даних і дії над ними.

Алфавіт Алфавітом називають сукупність символів, що використовуються в мові. В С ++ вони утворюють букви, цифри і спеціальні символи. Як букви використовуються прописні букви латинського алфавіту від *A* до *Z* та маленьку літери від *a* до *z*, а також знак підкреслювання _. Для десяткових цифр використовуються арабські цифри від 0 до 9. Спеціальні символи в мові С ++ застосовуються для різних цілей: від організації тексту програми до визначення вказівок компілятору. Спеціальні символи перераховані:



З символів алфавіту формуються лексеми мови [1]:

- ідентифікатори;
- знаки операцій;
- константи;
- роздільники.

Необхідно пам’ятати, що мова С є залежною від регістру, що означає, що велика і маленька літери є різними символами.

Лексеми мови. З зображуваних символів алфавіту формуються лексеми (tokens) мови. Лексеми – послідовності символів вихідного коду програми, мають певне смислове значення.

У мові С ++ є такі категорії лексем:

- ідентифікатори (identifier);
- ключові (зарезервовані) слова (keyword);
- знаки операцій (operator);
- константи - літерали (literal);
- роздільники (знаки пунктуації - punctuator).

Розглянемо ці лексичні елементи мови докладніше.

Ключові слова

Ключові слова - це зарезервовані ідентифікатори, які мають спеціальне значення для компілятора.

auto	continue	float	interrupt	short	unsigned
asm	default	for	long	signed	void
break	do	far	near	sizeof	volatile
case	double	goto	pascal	static	while
cdecl	else	huge	switch	struct	
char	enum	if	register	typedef	
const	extern	int	return	union	

Імена об'єктів програми не можуть співпадати з ключовими словами.

Ідентифікатори. Ідентифікатори використовуються як імена змінних, функцій і типів даних. Вони записуються за такими правилами.

1.Ідентіфікатори починаються з літери (знак підкреслення також є буквою).

2.Ідентіфікатор може складатися з латинських букв і цифр (прогалини, точки та інші спеціальні символи при написанні ідентифікаторів неприпустимі).

3.Между двома ідентифікаторами повинен бути, принаймні, хоча б один пропуск.

4.Ідентіфікатор може бути довільної довжини, але значущими є тільки перші 32 символу в середовищі WINDOWS і 8 символів в середовищі DOS; інші символи ігноруються.

Правильні ідентифікатори	Неправильні ідентифікатори
wiggly	$ A ^ **
cat	do not
HOT_key	HOT-key
_grab1	1grab

При написанні ідентифікаторів можна використовувати як прописні, так і малі літери. На відміну від інших мов програмування, компілятор мови С ++ розрізняє їх в запису ідентифікатора. Для більш простого читання і розуміння ідентифікаторів рекомендується використовувати імена ідентифікаторів, що складаються з малих і великих літер. Кожен ідентифікатор має тип, який встановлюється при його оголошенні. Компілятор мови С ++ не допускає використання ідентифікаторів, які збігаються з написання з ключовими словами. Наприклад, ідентифікатор auto неприпустимий (проте припустимо ідентіфікаторAUTO).

Константи **–** це величини, які не змінюється протягом виконання всієї програми.

Всі константи незалежно від типу даних можна поділити на дві категорії: іменовані константи і константи, які не мають власного імені (***літерали***). наприклад:

25 – константа цілого типу;

3.14 – дійсна константа;

'A' – символьна константа.

Літерали цілих типів можна записати в десятковому, вісімковому і шістнадцятковому вигляді. Ось як виглядає число 20, представлене десятковим, вісімковим і шістнадцятковим літералами:

20 // десятковий

024 // вісімковий

0x14 // шістнадцятковий

За замовчуванням всі цілі літерали мають тип signed int – ціле із наком. Можна явно визначити цілий літерал який має тип long, приписавши в кінці числа букву L (використовується як прописна L, так і рядкова l, однак для зручності читання не слід вживати малу: її легко переплутати з

1). Буква U (або u) в кінці визначає літерал як unsigned int, а дві букви - UL або LU - як тип unsigned long. наприклад:

128u 1024UL 1L 8Lu

В мові С розрізняють вбудовані або системні константи та визначені користувачем. Прикладом системної константи є значення π : pi.

Для визначення власних констант існує два механізми:

- за дпомогою препроцесорно директиви **`#define;`**
- за дпомогою ключового слова **`const.`**

Різниця між цими двома способами задання полягає в тому, що в першому випадку, коли під час виконання програми в тексті програми з’являється ім’я константи виконується підстановка значення, у другому випадку підстановка виконується на етапі обробки вихідного коду і програма працює швидше. Також для другого способу виконується перевірка правильності задання константи і у випадку помилки видається відповідне повідомлення.

Змінні – це дані, які можуть змінювати свої значення в процесі виконання програми.

Оператори та ***операнди***

Операнд – це спеціальна величина, яка обробляється в програмі. ***Оператор*** – це деяка дія, яке виконується з операндом. За аналогією з нашою мовою операнд – це підмет, а оператор – присудок.

В мові С розрізняють простий та складений оператор. Простий оператор завершується символом «;». Він може бути записаний в одному або декількох рядках, «;» є ознакою завершення.

Приклад.

1) a=3;

2) a

=

3;

Записи 1) і 2) визначають однаковий оператор присвоєння.

В одному рядку може бути записано декілька операторів, «;» є роздільником між ними.

Приклад.

a=2; в=3.5; – запис двох операторів в одному рядку.

Сукупність простих операторів, які заключні у фігурні дужки утворюють складений оператор.

Приклад.

{a=2; в=3.5;} – складений оператор.

В командах де передбачено виконання дії, а необхідно виконати декілька дій використовують складені оператори.

Приклад.

if (a>b) {a=2*a; b=-b;}

В даному прикладі, якщо умова a>b є вірною, виконується дві дії: a=2*a, b=-b.

Основні директиви препроцесора

Директива препроцесора це команди компілятора, які виконуються на початку програми.

Препроцесор – це програма, яка опрацьовує директиви.

Директиви препроцесора задаються за наступним шаблоном:

#ім'я_директиви параметри.

Директиви мови С розпочинаються символом # після якого без пропуску записується ідентифікатор директиви.

- З однією з деректив ми вже познайомились. Директива **#include** вставляє код із зазначеного файлу в поточний файл, тобто, просто підключивши інший файл, ми можемо користуватися його функціями, класами, змінними. Заголовки зазвичай знаходяться або в поточній директорії, або в стандартному системному каталозі.

Підключення заголовних файлів виконується під час компіляції, або як файл, який є частиною вашого проекту. Ця функція залежить від конкретної реалізації вашого компілятора, тому для отримання більш докладної інформації, покопатися в налаштуваннях свого компілятора. Якщо підключений файл не знайдений, процес компіляції завершується з помилкою.

Ім'я файлу, що підключається береться:

- у символи < > для підключення стандартних бібліотек;
- у лапки для підключення власних файлів.

Приклади.

#include <iostream> – підключення стандартної бібліотеки вводу-виводу через потоки.

#include "path-spec" – підключення файлу користувача з іменем path-spec.

- Директива #define – приймає дві форми:

- визначення констант;
- визначення макросів.

Визначення констант: **#define** ім’я_константи значення.

Приклади.

#define A 5 – константі A присвоюється значення 5.

#define B 2.72 – константі B присвоюється значення 2,72.

#define TEXT "Марс" – константі TEXT присвоюється значення Марс.

Визначення макросів: : **#define** ім’я_макроса(параметри) вираз.

Макрос, – це записана команда або послідовність звичайних команд, яка може бути відтворена для автоматизації часто повторюваних дій.

Приклади.

- **#define** SUM(a,b) (a)+(b) – обчислення суми двох чисел.
- **#define** MAX(num1, num2) ((num1) > (num2) ? (num1) : (num2)) – макрос для визначення найбільшого значення з двох чисел.

У наведених прикладад аргументи макроса беруться в дужки для коректного виконання дій (збереження знаку параметра).

- Директива **#undef** відміняє константу або макрос, раніше визначений за допомогою директиви **#define**: **#undef** ім’я_константи_або_макроса

Приклад.

```
#define E 2.71828 // визначення константи
int sumE = E + E; // звернення до константи E
#undef E // відміна константи
```

Прийнято ідентифікатори констант та макросів позначати великими літерами.

Існують і інші препроцесорні команди, але ми обмежимось розглядом основних.