## План лекцій

1. Основи мови C++
2. Керуючі інструкції C++
3. Вказівники, масиви, текстові рядки
4. Функції
5. Класи
6. Робота з об’єктами
7. Перевантаження операторів
8. Наслідування
9. Шаблони

Київський національний університет
Імені Тараса Шевченка
Фізичний факультет

# Програмування
## (мова C++)
## 1-й курс (1-й потік)

Лектор: Васильєв Олексій Миколайович
доктор фіз.-мат. наук, професор
кафедри теоретичної фізики

## Коротко:

1. C++ - мова ООП
2. Крім C++, популярними є мови Java та C#
3. C++ розрахована на певний тип процесора та операційної системи
4. Мова C++ є перехідною: можна створювати як звичайні, так і об'єктно-орієнтовані програми
4. Мови Java та C# створені для роботи в Internet
5. Мови Java та C# повністю об'єктно-орієнтовані

# Лекція 1. Основи мови C++

## Вступ. Історія появи C++

1. Мова B (Кен Томпсон)
2. Мова BCPL (Мартін Річардс)
3. Мова C (Деніс Рітчі, 1972) – розроблена програмістами і для програмістів
4. Мова C with classes (Б'єрн Страуструп, 1979) – вирішення проблеми удосконалення програмних кодів шляхом переходу на ООП
5. Мова C++ - з 1983 року
6. "Близькі" до C++ мови: Java та C#
7. "Швидкі" програми пишуть на C++, "універсальні" – на Java або C#

## Концепції програмування:

1. Процедурне програмування – “код, який діє на дані”
2. Об’єктно-орієнтоване програмування – “дані управляють доступом до коду”

## Переваги ООП:

1. Можна повторно використовувати створений раніше код
2. Програми добре структуровані
3. Програми легко тестувати
4. Програми легко розширюються (якщо необхідно)

# Основні принципи ООП

**ВСІ ООП-мови базуються на трьох механізмах**

## 1. Інкапсуляція:

механізм, який пов’язує та об’єднує в одне ціле код та дані. Для цього створюється об’єкт – конструкція, що підтримує інкапсуляцію. Базова одиниця інкапсуляції – клас. Клас містить дані та код для їх обробки. Об’єкт є екземпляром класа.

## 2. Поліморфізм:

механізм, що дозволяє використовувати один інтерфейс для різних дій. Дозволяє понизити рівень складності програм.

## 3. Наслідування:

Механізм, завдяки якому один об’єкт може отримувати властивості іншого об ’єкта. При цьому поліморфізм дозволяє змінювати поведінку дочірніх (похідних) класів

## Структура програми в С++:

1. Блок заголовків програми. На загал за допомогою інструкцій **#include** підключають зовнішні файли.

2. Блок з оголошенням класів.

3. Головний метод програми **main()**. Кожна програма містить один і тільки один метод **main()**.

4. Блок з описом функцій.

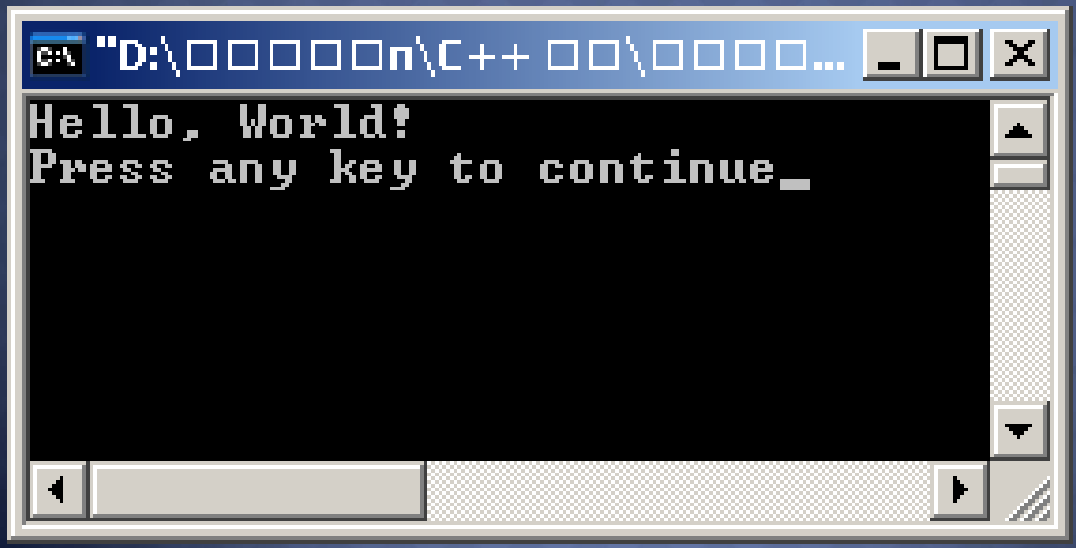
Обов’язковими є перший та третій блоки.

# Приклад простої програми на мові С++

**Програмний код:**

```
#include <iostream>
using namespace std;
int main(){
// Виводиться повідомлення
cout <<"Hello, World!\n";
return 0;}
```

**Результат:**



## Оголошення змінної:

1. Вказується ідентифікатор типу та ім’я змінної

2. Оголошення виконується в любому місті програми, але до першого використання

3. Декілька змінних можна оголошувати одразу (розділяються комами)

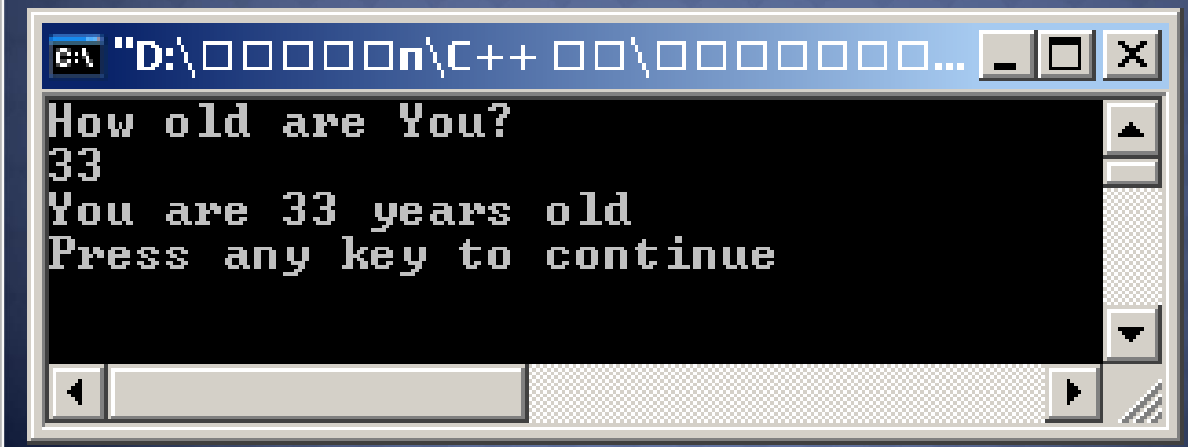
4. Оголошення змінної можна суміщати з ініціалізацією (присвоєнням значення)

# Приклад програми з оголошенням змінної

**Програмний код:**

```
#include <iostream>
using namespace std;
int main(){
int age;                                  // Змінна для запису віку
cout<<"How old are You?\n";    // Скільки Вам років?
cin >>age;                                // Треба вказати вік
cout<<"You are "<<age<<" years old\n";   // Результат
return 0;}
```

**Результат:**



## Динамічна ініціалізація змінної:

1. При оголошенні змінна ініціалізується на основі виразу, в який входять інші змінні і літерали

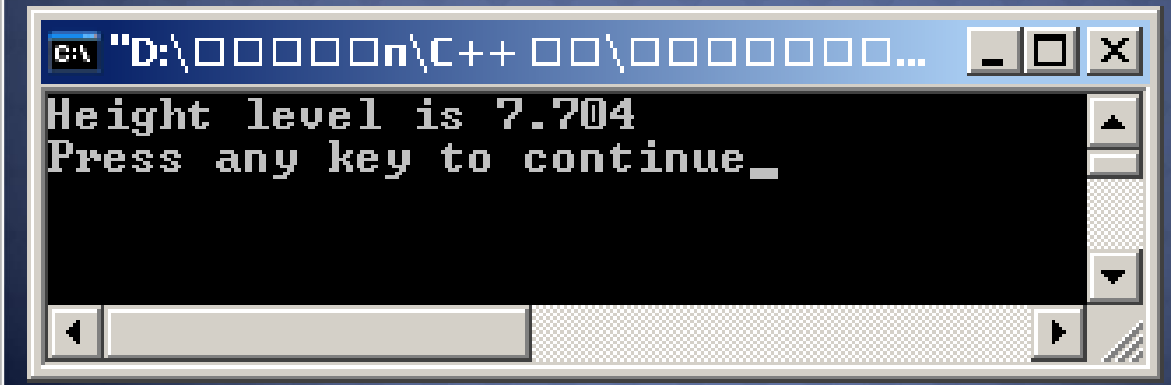
2. Вираз, яким визначається значення змінної при динамічній ініціалізації, має бути легітимним на момент використання

# Розрахунок висоти підйому тіла

**Програмний код:**

```
#include<iostream>
using namespace std;
int main(){
double v;          // Швидкість тіла - оголошення змінної
double t=1.2;     //Час польоту
double g=9.8;     // Прискорення вільного падіння
v=12.3;           // Швидкість - ініціалізація змінної
// Висота - динамічна ініціалізація змінної
double s=v*t-g*t*t/2;
cout<<"Height level is "<<s<<"\n";
return 0;}
```

**Результат:**



## Модифікатори типу:

1. Модифікатор **signed** - значення зі знаком

2. Модифікатор **unsigned** - значення без знака

3. Модифікатор **short** - скорочений тип

4. Модифікатор **long** - розширений тип

5. Всі чотири модифікатори використовуються з типом int
6. Модифікатори signed и unsignedвикористовують з типом char
7. Модифікатор long використовують з типом double

# Базові типи даних

| *Ідентифікатор типу* | *Тип даних* |
|---|---|
| `bool` | Логічний тип |
| `char` | Символьний тип |
| `wchar_t` | Символьний двухбайтовий тип |
| `double` | Дійсні числа подвійної точності |
| `float` | Дійсні числа |
| `int` | Цілі числа |
| `void` | Значення не повертається |

# Мінімальні діапазони даних

**(32-розрядне середовище)**

| *Тип* | *Кількість біт* | *Діапазон значень* |
|---|---|---|
| `bool` | 1 | Значення `true` або `false` |
| `char` | 8 | від `-128` до `127` |
| `wchar_t` | 16 | від `0` до `65535` |
| `double` | 64 | від `2.2E-308` до `1.8E+308` |
| `float` | 32 | від `1.8E-38` до `1.8E+38` |
| `int` | 32 | від `-2147483648` до `2147483647` |
| `unsigned char` | 8 | від `0` до `255` |
| `signed char` | 8 | від `-128` до `127` |
| `signed int` | 32 | від `-2147483648` до `2147483647` |
| `short int` | 16 | від `-32768` до `32767` |
| `unsigned short int` | 16 | від `0` до `65535` |

# Мінімальні діапазони даних (продовження)

**(32-розрядне середовище)**

| *Тип* | *Кількість біт* | *Діапазон значень* |
|---|---|---|
| `signed short int` | 16 | від `-32768` до `32767` |
| `long int` | 32 | від `-2147483648` до `2147483647` |
| `unsigned long int` | 32 | від `0` до `4294967295` |
| `signed long int` | 32 | від `-2147483648` до `2147483647` |
| `double` | 64 | від `2.2E-308` до `1.8E+308` |
| `float` | 32 | від `1.8E-38` до `1.8E+38` |
| `long double` | 64 | від `2.2E-308` до `1.8E+308` |

## Оператори в мові С++:

1. Арифметичні оператори
2. Оператори порівняння
3. Логічні оператори
4. Опобітові оператори

# Арифметичні оператори

| *Оператор* | *Призначення* |
|---|---|
| + | Додавання |
| – | Віднімання |
| * | Множення |
| / | Ділення. Якщо операндами є цілі числа, виконується ділення націло |
| % | Залишок від ділення |
| ++ | Інкремент |
| -- | Декремент |

**Скорочені форми операторів:**

**формат** `змінна1 оператор=вираз (напр., x+=z)`

**те саме що** `змінна=змінна оператор вираз (x=x+z)`

**Префіксна і постфіксна форми оп-в інкремента та декремента:**

`n++ або ++n (n=n+1) m-- або –m (m=m-1)`

**Оператори присвоєння і рівності:**

```
int a=5,b=6;
a==b; //логічне false
a=b;   //логічне true
```

**Логічне виключаюче АБО (XOR) :**

A XOR B або

(A||B)&&!(A&&B)

## Оператори порівняння

| *Оператор* | *Призначення* |
|---|---|
| > | Більше |
| < | Меньше |
| >= | Більше або рівно |
| <= | Меньше або рівно |
| == | Рівно |
| != | Не рівно |

## Логічні оператори

| *Оператор* | *Призначення* |
|---|---|
| && | Логічне *І (AND)* |
| \|\| | Логічне *АБО (OR)* |
| ! | Логічне *НІ (NOT)* |

**Приклади побітових операцій:**

5&3 = 1
5|3 = 7
5^6 = 3
~5 = -6
5>>2 = 1
5<<2 = 20

# Побітові оператори

| *Оператор* | *Призначення* |
|---|---|
| & | Побітове *І (AND)* |
| \| | Побітове *АБО (OR)* |
| ^ | Побітове *виключаюче АБО (XOR)* |
| ~ | Побітове заперечення *НІ (NOT)* |
| >> | Зсув вправо. Старший біт залишається незмінним |
| << | Зсув вліво. Молодші біти заповнюються нулями, старші втрачаються |

# Оператор присвоєння і приведення типів

**Синтаксис використання:** `змінна=вираз`

**Особливості:**

**1. В мові С++ оператор присвоєння повертає значення**

```
x=y=z=3;
n=(m=6)+3;
```

**2. Виконується автоматичне приведення типів (тип виразу трансформується до типу змінної, при цьому можуть втрачатись дані)**

```
int a;
double x=3.5;
a=x;  //змінна а отримує значення 3
```

**3. Можна виконувати явне приведення типу. Для цього перед виразом у круглих дужках вказують кінцевий тип**

```
5/3 ;              //результат 1
(double)5/3;    //результат 1.66666
```

# Тернарний оператор

1. У оператора 3 аргументи

2. Синтаксис виклику: **умова?команда1:команда2;**

Якщо справедлива ***умова***, виконується ***команда1***. Інакше виконується ***команда2***.

3. Оператор повертає результат – це результат виклику відповідної команди

4. Приклад використання:

```
#include <iostream>
using namespace std;
int main(){
int n;
double x;
cout<<"Enter  n = ";
cin>>n;
x=n>0?5.4:3.2;
cout<<"x = "<<x<<"\n";
return 0;}
```

# Резюме

**1.** Мова програмування C++ є розширенням мови C для підтримки об’єктно-орієнтованої парадигми. В C++ може бути реалізований як процедурний, так і ООП-підхід.

**2.** Програма в C++ містить тільки один метод **main()**, виконання якого ототожнюється з виконанням програми.

**3.** При оголошенні змінних дл них вказується тип та ім’я. Для оголошення констант використовують ключове слово const. Змінні слід оголосити і ініціалізувати до того, як вони використовуються в програмі.

**4.** В C++ існує сім базових ідентифікаторів типів: **int** (цілі числа), **float** (дійсні числа), **double** (дійсні числа подвійної точності), **char** (символи), **bool** (логічний тип) та **void** (відсутність результату у функції). За допомогою модифікаторів типу параметри азових типів можна змінювати.

# Резюме (продовження)

**5.** Всі оператори  C++ можна розбити на 4 групи: арифметичні, логічні, оператори порівняння та побітові оператори.

**6.** В C++ існує тернарний оператор – спрощена форма умовного оператора.

**7.** В якості оператора присвоєння використовується знак рівності (тобто **=**). Оператор присвоєння повертає результат (значення виразу справа від оператора).

**8.** В C++ виконується автоматичне приведення типів. Приведення типів можна також виконувати і в явному вигляді.