

Основи програмування

Лектор

к.т.н. доцент кафедри програмних систем і технологій

КНУ ім. Тараса Шевченка

Ковалюк Тетяна Володимирівна

tkovalyuk@ukr.net

tetyana.kovalyuk@knu.ua

Лекція 3

Елементи мов С/С++

Лектор к.т.н. доцент кафедри програмних систем і технологій
КНУ ім. Тараса Шевченка
Ковалюк Тетяна Володимирівна

tkovalyuk@ukr.net

Зміст



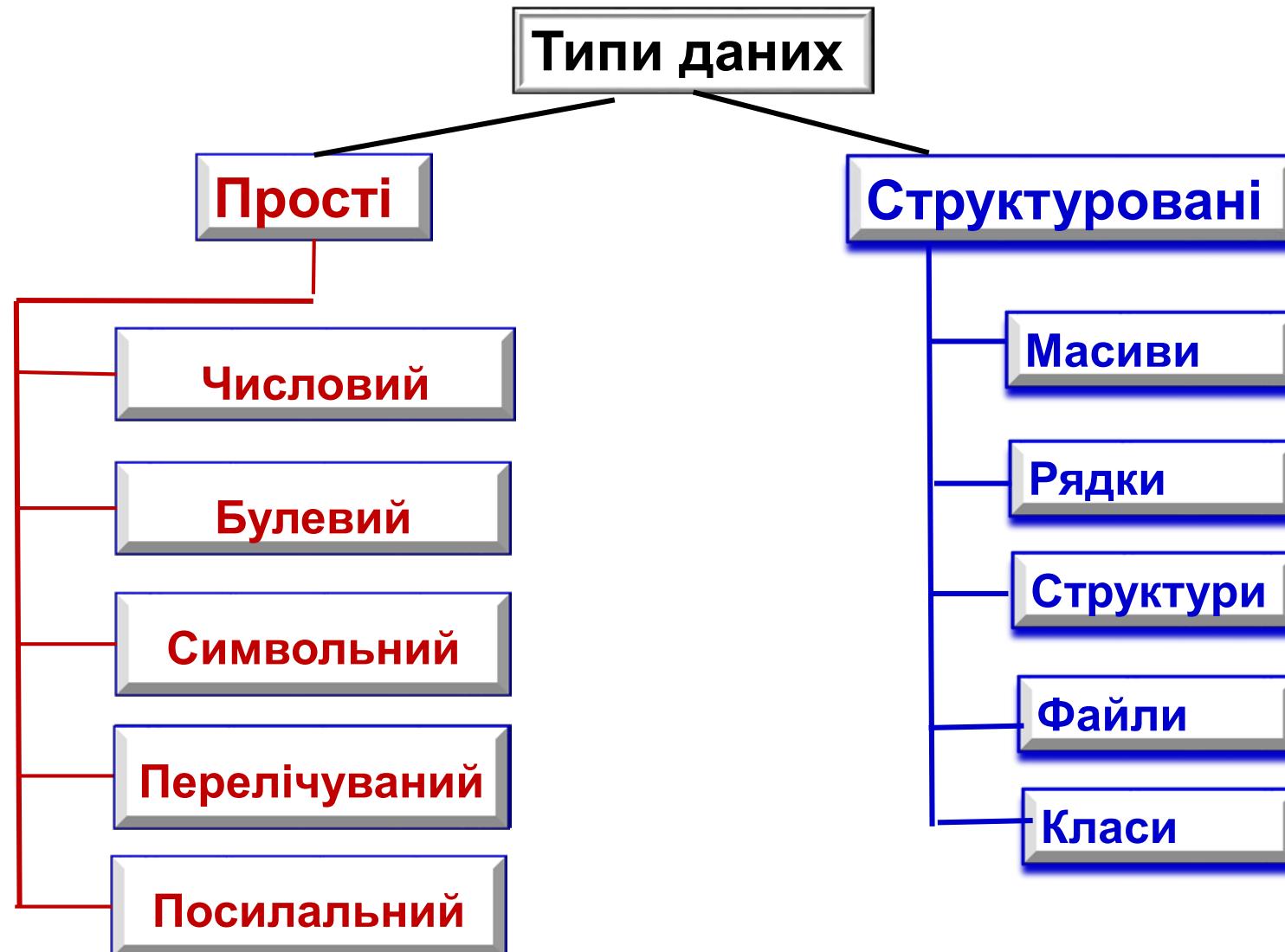
- 2.1. Робота у середовищі Visual Studio .NET
- 2.2. Словник мови C/C++ та загальна структура програми
- 2.3. Прості типи даних**
- 2.4. Константи, змінні, вирази**
- 2.5. Операції присвоєння та функції введення-виведення**

Поняття типу даних

Тип даних визначає:

1. Множину допустимих значень, яких може набувати змінна або константа зазначеного типу;
2. Множину допустимих операцій, що застосовуються до даних певного типу;
3. Спосіб зображення даних у пам'яті комп'ютера.

Різновиди типів даних в С/С++



Прості типи даних

Операції з даними

- Операції — це дії, що виконуються над певними значеннями.
- Значення, над яким здійснюється операція, називають її *операндом*.
- Залежно від типів операндів операції поділяють на
 - арифметичні,
 - логічні,
 - посилальні,
 - рядкові.
- Відповідно до кількості операндів операції в мовах C/C++ поділяють на
 - унарні
 - бінарні.

Цілочислові типи даних

Цілочислові типи — це типи даних, множини допустимих значень яких є множинами цілих чисел.

Ідентифікатор типу	Кількість байтів	Діапазон значень
char	1	-128 .. +127
unsigned char	1	0 .. 255 ($2^8 - 1$)
short	2	-32 768 .. 32 767 ($2^{15} - 1$)
unsigned short	2	0 .. 65 535
int	4	-2 147 483 648 .. 2 147 483 647 ($2^{31} - 1$)
unsigned int	4	0 .. 4 294 967 295
long	4	-2 147 483 648 .. 2 147 483 647 ($2^{31} - 1$)
unsigned long	4	0 .. 4 294 967 295

Операції над цілочисловими типами

Знак операції	Зміст операції	Приклади застосування та результати
+	Додавання	$1 + 2 = 3;$ $1 + 32\ 767 = -32\ 768$ (переповнення комірки для типу short); $-32\ 768 + (-32\ 768) = 0$
-	Віднімання	$1 - 2 = -1;$ $-32\ 768 - 1 = 32\ 767$; (переповнення комірки для типу short); $32\ 768 - (-32\ 768) = 0$
*	Множення	$2 * 2 = 4; 256 * 128 = -32\ 768$ (переповнення комірки для типу short); $256 * 256 = 0; 32\ 767 * 32\ 767 = 1$
/	Визначення цілої частини від ділення	$7 / 3 = 2; -7 / 3 = -2; 7 / -3 = -2; -7 / -3 = 2$
%	Визначення остачі від ділення	$7 \% 3 = 1; -7 \% 3 = -1;$ $7 \% -3 = 1; -7 \% -3 = -1$
-(унарний)	Зміна знаку числа	$-(1) = -1;$ $-(-32\ 768) = -32\ 768$ (переповнення комірки для типу short)

Операції над цілочисловими типами

Знак операції	Зміст операції	Приклади застосування та результати
<code>++</code>	Префіксний та постфіксний інкремент (збільшення)	<code>a=1;</code> <code>a++;</code> <code>++a; (результат a=2)</code>
<code>--</code>	Префіксний та постфіксний декремент (зменшення)	<code>a=1;</code> <code>a--;</code> <code>--a; (результат a=0)</code>
<code><<</code>	Зсув бітів ліворуч	<code>2<<1 = 4; 1<<5 = 32;</code>
<code>>></code>	Зсув бітів праворуч	<code>2>>1 = 1; 8>>2 = 2;</code>

Дійсні типи даних

- ❑ Множина допустимих значень будь-якого дійсного типу є скінченою підмножиною множини раціональних чисел і містить, зокрема, усі цілі числа типу `int`.
- ❑ Для запису дійсних чисел в оперативній пам'яті використовується **формат з плаваючою точкою**.
- ❑ Дійсне число у форматі з плаваючою точкою має **мантису та порядок**.
- ❑ Кількість цифр у мантисі характеризує **точність числа**.
- ❑ Чим більше цифр у мантисі, тим вище точність.
- ❑ Порядок визначає величину числа, тобто справжнє місце знаходження десяткової точки в числі.

Ідентифіатор типу	Кількість байтів	Найменше за модулем число	Найбільше за модулем число
<code>float</code>	4	$3.4 \cdot 10^{-38}$	$3.4 \cdot 10^{38}$
<code>double</code>	8	$1.7 \cdot 10^{-308}$	$1.7 \cdot 10^{308}$
<code>long double</code>	10	$3.4 \cdot 10^{-4932}$	$1.1 \cdot 10^{4932}$

Операції для дійсних типів даних

- ❑ Для дійсних типів означенено чотири арифметичні операції:
 - додавання (+),
 - віднімання (-),
 - множення (*),
 - ділення (/).
- ❑ Виконання унарних операцій **інкремента** «++» та **декремента** «--» збільшує або зменшує на одиницю значення цього числа.
- ❑ Також над даними дійсних типів можна виконувати ті самі операції порівняння, що і над даними цілих типів.

Логічний (булів) тип даних

- ❑ Множина допустимих значень булевого, або логічного, типу містить дві константи: `false` (хибність) і `true` (істина).
- ❑ Ідентифікатором логічного типу є слово `bool` (для C++).
- ❑ Булевим може бути будь-яке арифметичне значення.
- ❑ У C/C++ усе, що `відмінне від нуля`, вважається `істинним`.

Булеві операції

A	B	A && B	A B	! A
<code>false</code>	<code>false</code>	<code>false</code>	<code>false</code>	<code>true</code>
<code>false</code>	<code>true</code>	<code>false</code>	<code>true</code>	<code>true</code>
<code>true</code>	<code>false</code>	<code>false</code>	<code>true</code>	<code>false</code>
<code>true</code>	<code>true</code>	<code>true</code>	<code>true</code>	<code>false</code>

Логічне множення Логічне додавання Логічне заперечення

Приклад. Обчислити вираз 6 & 3.

У двійковій системі числа 6 та 3 мають вигляд **110** і **011** відповідно . Доожної пари розрядів цих операндів застосовано операцію &:
 $110_2 \& 011_2 = 010_2$. Двійковий результат 010 дорівнює десятковому числу 2. Отже, **$6_{10} \& 3_{10} = 2_{10}$**

Символьний тип

- Множина допустимих значень *символьного* (літерного) типу — це множина *цілих чисел* в діапазоні від –128 до 127.
- Зберігання значення такого типу потребує одного байта оперативної пам'яті.
- Найчастіше такий тип застосовується для зберігання символів кодової таблиці ASCII, а отже, даними цього типу є **окремі символи**.
- Кожному символу відповідає ціле число (код) в діапазоні від 0 до 127.
- Ці коди ідентичні на всіх IBM-сумісних комп'ютерах.
- Символи з кодами від 0 до 31 належать до керуючих символів (ESC-послідовності).
- ESC-послідовності записують лексемою виду '**\symbol**'

Символьний тип

Символи з кодами від 0 до 31 належать до керуючих символів.

Керуючі послідовності записують лексемою виду '\symbol'

Деякі керуючі символи таблиці ASCII кодів (ESC- послідовності)

Код	Назва символу	Запис у програмі
7	Звуковий сигнал	'\a'
8	Пробіл	'\b'
9	Горизонтальна табуляція	'\t'
10	Переведення рядка	'\n'
12	Переведення сторінки	'\f'
13	Повернення каретки	'\r'

Операції над даними символьного типу

1. Операції порівняння.

Символи вважають рівними, якщо рівні їх ASCII-коди. Один символ вважають більшим за інший, якщо його ASCII-код більший. Зокрема,

'0' < '1' < ... < '9' < 'A' < 'B' < ... < 'Z' < 'a' < 'b' < ... < 'z'.

2. Операції додавання та віднімання., у результаті виконання яких утворюються нові символи. Ці операції позначають символами «+» та «-».

Наприклад, '**1**'+'**2**' = '**c**', оскільки символ '**1**' має код 49, символ '**2**' — код 50, а символ '**c**' — код 99 (49+50).

Перелічуваний тип даних

- ❑ Перелічуваний тип означується користувачем.
- ❑ Такий тип задається переліком усіх елементів множини допустимих значень.
- ❑ Кожне значення іменується певним ідентифікатором і зазначається у списку, який береться у **Фігурні дужки**.
- ❑ Ідентифікатор перелічуваного типу треба оголосити у програмі з ключовим словом **enum**.

Синтаксис оголошення перелічуваного типу:

```
enum ідентифікатор типу {ідентифікатор_1, ідентифікатор_2, ...,  
ідентифікатор_n};
```

Тут **ідентифікатор типу** — рядок символів;

ідентифікатор_1, ..., ідентифікатор_n — допустимі значення перелічуваного типу, задані рядковими константами, які отримали назву **констант перелічуваного типу**.

Перелічуваний тип даних

Приклад означення ідентифікатора перелічуваного типу:

```
enum WorkWeek {Mon, Tue, Wed, Thu, Fri, Sat, Sun };
```

```
enum Color {red,green,blue};
```

```
enum WinterMonth {December,January,February};
```

Оголошення змінної перелічуваного типу

Ідентифікатор_типу ідентифікатор_змінної;

Константи. Змінні. Вирази

- Константа в тексті програми С/C++ позначається:
 - ідентифікатором.
 - безпосереднім значенням;
- **Числові константи** — це цілі десяткові, шістнадцяткові та вісімкові числа, а також дійсні десяткові числа.
- **Символьна константа** — це ASCII-символ, записаний в одинарних лапках (апострофах), або символ, записаний як ESC-послідовність: 'A', '\n'. Символьна константа має тип **char**.
- **Рядкова константа** являє собою послідовність символів, записаних у подвійних лапках: "Visual C++ .NET".
- **Логічні константи** мають тип **bool** і значення **false** та **true**, що означає хибність та істинність відповідно.

Різновиди констант

- ❑ Іменовані константи — це об'єкти, оголошені з ключовим словом **const**.
- ❑ Оскільки константі не можна присвоїти значення, її слід ініціалізувати на початку виконання програми.
- ❑ Константу можна визначити за допомогою директиви препроцесора

const тип ідентифікатор = вираз;

```
const int year = 2004;           //ціличеслова константа
const double price = 250.56;     //дійсна константа
const char answer = 'y';        //символьна константа
const char bell = '\a';         //символьна константа
const bool flag = false;        //булева константа
const char* str = "pointer to string"; //покажчик на
                                         //константний рядок
const char title[] = "C++ Programming Language";
```

#define ідентифікатор вираз

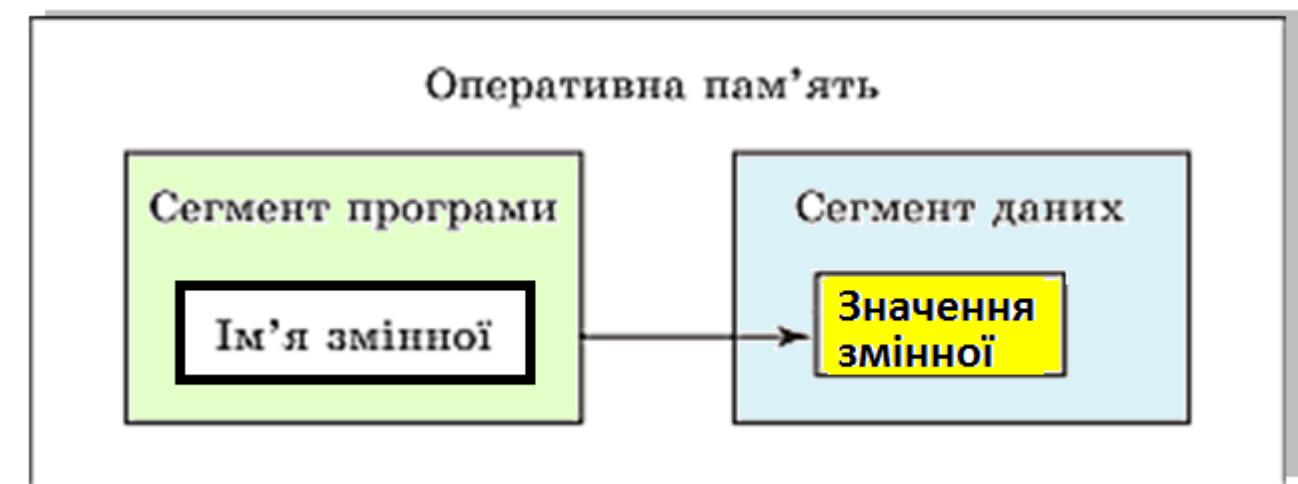
```
#define digit 1000+200          //ціличеслова константа
#define RealConst = 3.14          //дійсна константа
#define SymbolConst = '@'         //символьна константа
#define EscKey = 27                //ціличеслова константа
#define StringConst = "section of constant" //рядкова
                                         // константа
#define BoolConst = false         //булева константа
```

Змінні

- Змінна величина — це узагальнення, абстракція якогось реального чи уявного об'єкта, що може перебувати в різних станах.
- Змінні набувають різних значень під час виконання програми.
- Змінні виконують функцію зберігання даних.
- Змінні зберігають свої значення в комірках оперативної пам'яті (RAM)

Синтаксис оголошення змінної

тип ідентифікатор;



Приклад використання змінних



```
typedef char* pointer; //перейменування типу
//означення перелічуваних типів
enum WorkWeek {Mon, Tue, Wed, Thu, Fri, Sat, Sun};
enum WinterMonth {December, January, February};

char* string; //змінна типу покажчика
float a, b, c; //дійсні змінні
int i, j; //цілочислові змінні
bool tag; //булева змінна
char key; //символьна змінна
WorkWeek day; //змінна перелічуваного типу
WinterMonth cold; //змінна перелічуваного типу
```

Область видимості змінних

Змінні можна оголосити в різних частинах програми:

- у функціях;
- в окремих блоках функції;
- в області глобальних оголошень поза межами функцій;
- в заголовному файлі, який потрібно підключити до програми за допомогою директиви препроцесора `#include`.

Залежно від того, де оголошена змінна, їй надається **область видимості, тобто область, де її можна використовувати.**

Вирази

Вираз є послідовністю операцій, **операндами** яких можуть бути **змінні**, константи, виклики функцій та інші вирази.

Для керування порядком виконання операцій застосовуються круглі дужки.



Вираз включає: змінні, константи, виклики функцій, операції

Приклад виразу мовою C/C++

`x= (a + b) / (c - d) + sin(a / b) * log(a) / abs(b - c);` →

Математичний вираз

$$x = \frac{(a+b)}{(c-d)} + \frac{\sin \frac{a}{b} * \ln a}{|b-c|}$$

Деякі математичні функції у виразах

Модуль цілого числа	<code>abs(x)</code>	Десятковий логарифм	<code>log10(x)</code>
Модуль дробового числа	<code>fabs(x)</code>	Піднесення е до степеню x	<code>exp(x)</code>
Синус	<code>sin(x)</code>	Піднесення x до степеню y	<code>pow(x,y)</code>
Косинус	<code>cos(x)</code>	Піднесення 10 до степеню x	<code>pow10(x)</code>
Тангенс	<code>tan(x)</code>	Квадратний корінь	<code>sqrt(x)</code>
Арксинус	<code>asin(x)</code>	Округлення вгору	<code>ceil(x)</code>
Арккосинус	<code>acos(x)</code>	Округлення вниз	<code>floor(x)</code>
Арктангенс	<code>atan(x)</code>	Остача від ділення x на y	<code>fmod(x,y)</code>
Натуральний логарифм	<code>log(x)</code>		

Для використання у програмі математичних функцій потрібно приєднати до програми заголовний файл `<math.h>` або `<cmath>`

Усі наведені функції, крім `abs(x)` і `pow10(x)`, мають тип аргументу і результату `double`.

Для функцій `abs(x)` і `pow10(x)` тип аргументу і результату є `int`

Правила запису математичних виразів

- Кількість відкритих і закритих дужок у виразах повинна бути однаковою.
- Усі елементи виразів (дроби, показник степеню, індекси) записують у горизонтальному рядку
- Вирази можна записувати у декількох рядках. Розривати вирази можна, наприклад, після символу арифметичної операції. Сам символ дублювати не потрібно.

Операції у виразах

Операція	Назва		Приклад застосування
Арифметичні операції			
+	Додавання		expr + expr
-	Віднімання		expr - expr
*	Множення		expr * expr
/	Ділення		expr / expr
%	Залишок від ділення цілих чисел (ділення за модулем)		expr % expr
Логічні операції			
&&	I (логічне множення)		expr && expr
	АБО (логічне додавання)		expr expr
!	НЕ (заперечення)		! expr
? :	Імплікація (арифметичний оператор if)		expr ? expr : expr
Відношення (порівняння)			
==	Дорівнює		expr == expr
!=	Не дорівнює		expr != expr
>	Більше		expr > expr
<	Менше		expr < expr
>=	Не менше		expr >= expr
<=	Не більше		expr <= expr
Бітові операції			
&	I (кон'юнкція)		expr & expr (7&5 = 5)
	Включене АБО (диз'юнкція)		expr expr (7 5 = 7)
^	Виключене АБО		expr ^ expr (7^5 = 2)
~	Заперечення		~expr (~6 = 1)

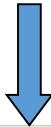
Операції у виразах

Операція	Назва	Приклад застосування
Унарні операції		
<<	Зсув заданої кількості бітів ліворуч	expr << expr
>>	Зсув заданої кількості бітів праворуч	expr >> expr
++	Інкремент	++ lvalue та lvalue ++
--	Декремент	-- lvalue та lvalue --
	Змінення знаку значення виразу	
&	Адреса об'єкта	& lvalue
*	Розіменування	* expr
new	Створення (розміщення) об'єкта в пам'яті	new type
delete	Знищення (звільнення) об'єкта в пам'яті	delete pointer
()	Перетворення типів	(type) expr
[]	Індексація	pointer[expr]
()	Виклик функції	expr(expr_list)
sizeof	Розмір об'єкта	sizeof expr
sizeof	Розмір типу	sizeof(type)

Операції у виразах

Кожна змінна має унікальну адресу, за якою зберігає своє значення.

Отримати адресу змінної можна за допомогою операції **адресації (&)**.



Якщо **variable** — ім'я змінної, то вираз **&variable** визначить адресу змінної.

Коли адреса змінної відома, знайти значення, що зберігається за цією адресою, можна за допомогою операції **розіменування**.



Якщо **&variable** — адреса змінної **var**, то вираз ***(&variable)** визначить уміст цієї адреси.

Операції у виразах

□ **Унарні операції та операції присвоєння правоасоціативні**, тобто

вираз $a=b=c$ означає $a=(b=c)$,

вираз $*p++$ означає $*(p++)$.

Решта операцій — **лівоасоціативні**, наприклад, вираз $a+b-c$ обраховується $(a+b)-c$.

□ **Операція імплікації ?: max = (a < b) ? b : a;**

Слід читати: якщо $a < b$, тоді $\text{max}=b$, інакше $\text{max}=a$

□ **Операція перетворення типів** для явного перетворення типів
(type)expression

Наприклад,

```
int count=10;  
float price=5.25;  
double sum = (double)count * price;
```

Правила визначення пріоритету операцій

Пріоритет операцій у мові C/C++

Групи операцій	Операції			
Унарні	- (змінення знаку), ++, --, !, &			
Мультиплікативні	*	/	&&	%
Адитивні	+, -			^
Зсуву	<<, >>			
Відношення	==, !=, <, >, <=, >=			

1. Операнд, що міститься між двома операціями з різними пріоритетами, зв'язується з операцією, яка має **вищий пріоритет**;
2. Операнд, що міститься між двома операціями з рівними пріоритетами, зв'язується з операцією, яка записана **ліворуч**;
3. Вираз, який взято в дужки, обчислюється в першу чергу і далі розглядається як окремий операнд;
4. Операції з однаковим пріоритетом виконуються **зліва направо**.

Константні вирази

Константні вирази обчислюються під час компіляції програми, а не під час її виконання.

```
const float DigitAndFloat=1000+123.45; //константні вирази  
const int NumericalExpression = 976 – 453;  
const char CharExpression = 'a ' + 'b';  
const bool BoolExpression = 5 & 3;  
const bool ReferExpression = digit < NumericalExpression;  
const float expr=abs(floor(sin(3.14159265358979323846/2)+pow(2.0,5)));
```

Операція виклику функції

Функція – це іменована частина програми, яка реалізує деякий алгоритм, і яку можна багатократно викликати з будь-якого місця програми, якщо це не заборонено синтаксисом.

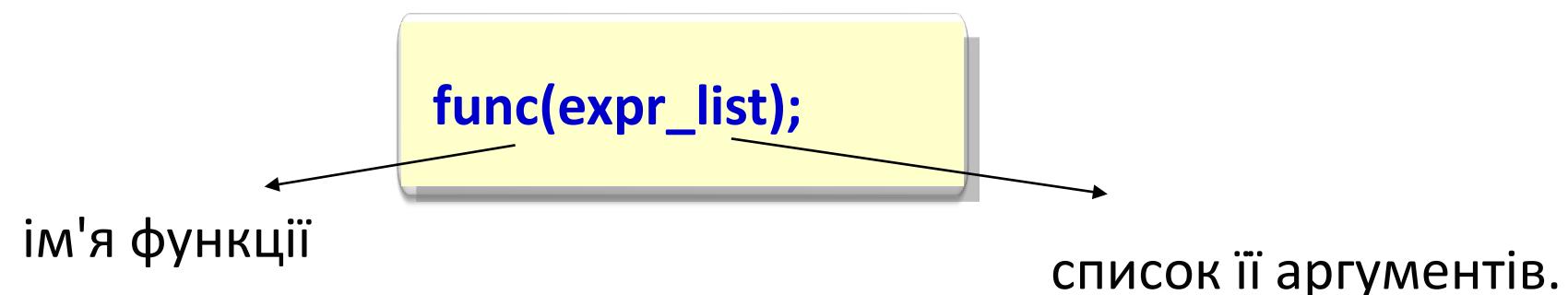
Виклик функції приводить до виконання вказаних у функції дій.

Операція виклику функції позначається круглими дужками ()

Для компіляції викликів компілятору достатньо мати інформацію про:

- ім'я функції,
- кількість параметрів
- типи параметрів,
- тип результату.

Операції виклику функції



Операції присвоєння

Синтаксис операції присвоєння:

ідентифікатор = вираз;

Алгоритм виконання операції присвоєння :

1. Обчислити значення виразу, записаного праворуч від символу присвоєння **=**.
2. Занести в комірку RAM, яка виділена для змінної, обчислене значення. Це означає, що отримане значення виразу надається змінній, позначеній ім'ям ліворуч від символу присвоєння **=**.

Процес модифікації значень змінних

```
//ex4.cpp Присвоєння
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
    int x, y, w;
    x = 1;
    y = 3;
    w = x + y;
    cout<<"w1: "<<w<<endl;
    w = w + 10;
    cout<<"w2: "<<w<<endl;
    system("pause");
}
```

Таблиця 2.9. Покрокове виконання програми ex2_3

Оператор, що виконується	Значення змінних (уміст комірок оперативної пам'яті)		
	x	y	w
Початок програми	-858993460	-858993460	-858993460
x = 1	1	-858993460	-858993460
y = 3	1	3	-858993460
w = x + y	1	3	4
w = z + 10	1	3	14



Комбіновані операції присвоєння

Таблиця 2.10. Комбіновані операції присвоєння

Операція	Назва	Приклад застосування
=	Занесення значення виразу до комірки оперативної пам'яті	<code>lvalue = expr</code>
+=	Додавання та присвоєння	<code>lvalue += expr</code>
-=	Віднімання та присвоєння	<code>lvalue -= expr</code>
%=	Взяти за модулем та присвоїти	<code>lvalue %= expr</code>
*=	Множення та присвоєння	<code>lvalue *= expr</code>
/=	Ділення та присвоєння	<code>lvalue /= expr</code>
>>=	Зсув бітів праворуч (ліворуч) та присвоєння	<code>lvalue >>= expr</code>
<<=		<code>lvalue <<= expr</code>
&=	Логічне бітове множення та присвоєння	<code>lvalue &= expr</code>
=	Логічне бітове додавання та присвоєння	<code>lvalue = expr</code>
^=	Виключене АБО та присвоєння	<code>lvalue ^= expr</code>

Комбіновані операції присвоєння

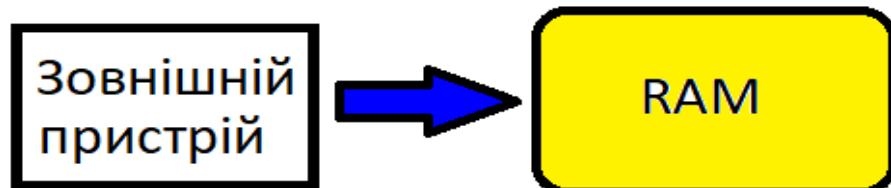
```
int total=10, item=2;  
total+=item;           // результат 12  10+2  
total-=item;           // результат 10  12-2  
total*=item;           // результат 20  10*2  
total/=item;           // результат 10  20/2  
total<<=item;          // результат 20  10*2  
total>>=item;          // результат 10  20/2  
total&=item;           // результат 2   1010&0010  
total|=item;            // результат 2   0010 | 0010
```



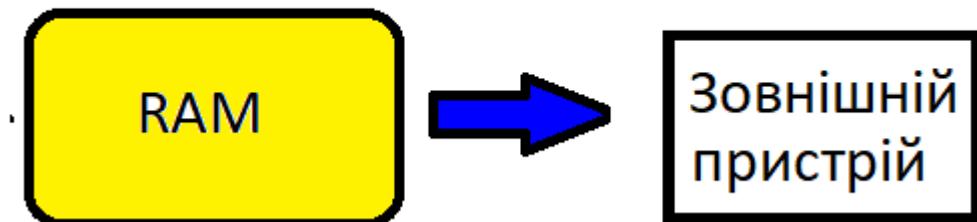
Основи введення-виведення в С/С+ +

- **Уведення даних** — це процес їх передавання із зовнішніх носіїв інформації або пристройів уведення даних до комірок оперативної пам'яті для їх подальшої обробки.
- **Виведення даних** — процес їх передавання з оперативної пам'яті на зовнішній носій інформації або пристрій виведення даних.
- **Буфер** — це область пам'яті для тимчасового зберігання даних.

Уведення даних



Виведення даних



Функції введення та виведення

Функція	Формат введення та виведення	Призначення	Приклад	Заголовний файл	Результат, що повертається функцією
_cprintf()	<code>int _cprintf (формат, арг1, арг2, ...);</code>	Виведення на консоль значень аргументів згідно з форматом виведення.	<code>int c; _cprint("%d",c);</code>	conio.h	Значення типу int, дорівнює кількості виведених символів
_cscanf()	<code>int _cscanf (формат, арг1, арг2, ...);</code>	Введення з консолі значень змінних згідно з форматом і присвоєння значень аргументам. Аргументами є адреси змінних в оперативній пам'яті.	<code>int a; char NAME[5]; _cscanf ("%s", Name); _cscanf ("%d", &a);</code>	conio.h	Значення типу int дорівнює кількості змінних, що отримали значення. При невдалому скануванні повертається значення EOF
_cputs()	<code>int _cputs (const char *str);</code>	Виведення на консоль рядка символів. Не додається символ кінця рядка '\0'. Borland C++ використовує cputs()	<code>char NAME[5]; _cputs(NAME); _cputs("Hello");</code>	conio.h	У випадку успіху повертається 0
_cgets()	<code>char *_cgets (char *str);</code>	Введення з консолі послідовності символів до символу "кінець рядка" ('\0')	<code>char a[5]; _cgets(a);</code>	conio.h	Значення типу int, що дорівнює покажчику на символ з індексом 2 (str[2]).
_getch()	<code>int _getch(void);</code>	Введення символу з консолі без відображення на екрані. Не використовується для Win32	<code>int c; c=_getch();</code>	conio.h	Код введеного з клавіатури символу
putch()	<code>int putch(int c);</code>	Виведення на консоль символу. Не використовується для Win32	<code>putch('B'); putch(_getch());</code>	conio.h	Повертається надрукований символ або значення EOF при невдалому виведенні.
getchar()	<code>int getchar(void);</code>	Введення символу із стандартного входного файлу stdin.	<code>int ch; ch=getchar();</code>	stdio.h або cstdio	Повертається код введеного символу або значення EOF
putchar()	<code>int putchar(int c);</code>	Повертається символ або значення EOF при невдалому виводу	<code>putchar(_getch());</code>	stdio.h або cstdio	Повертається символ або значення EOF при невдалому виводу

Функції введення та виведення

Функція	Формат введення та виведення	Призначення	Приклад	Заголовний файл	Результат, що повертається функцією
gets_s()	char *gets_s (char *s);	Введення із стандартного вхідного файла stdin рядка і розміщення його по показчику *s	char line[80]; gets(line);	stdio.h або cstdio	Покажчик на перший символ рядка або NULL при невдалому введенні
puts()	int puts (const char *s);	Виведення у стандартний вихідний файл stdout рядка і доповнення його символом нового рядка '\n'	char b[80]; puts(b);	stdio.h або cstdio	Повертається невід'ємне значення або EOF при невдалому виведенні
printf()	int printf (const char *format, argument, ...);	Виведення у стандартний вихідний файл stdout значень аргументів згідно з форматом виведення	int a; float b; char str[10]; printf("%d %f %s", a,b,str);	stdio.h або cstdio	Повертається кількість виведених байтів або від'ємне значення при невдалому виведенні
scanf_s()	int scanf_s (const char *format, address, ...);	Введення із стандартного вхідного файла значень змінних згідно з форматом і розміщення їх за вказаними адресами у оперативній пам'яті.	char AME[20]; int a; scanf("%s", AME); scanf("%d", &a);	stdio.h або cstdio	Повертається кількість змінних, що отримали значення. При невдалому скануванні повертається значення EOF
sprintf()	int sprintf (char *str, const char *format, argument, ...);	Розміщення у рядку символів значень аргументів із пам'яті згідно з шаблоном. Використовується для перетворення типів даних	char str[3]; sprintf(str,"%d", 13);	stdio.h або cstdio	Повертається кількість виведених символів.
sscanf_s()	int sscanf (рядок, фор-мат, арг1, ...);	Читання із рядка у адреси пам'яті згідно з форматом шаблонів. Використовується для перетворення типів	int a; float b; char str[5]; sscanf(str,"%d%", &a, &b);	stdio.h або cstdio	Повертається кількість введених символів або EOF при невдалому введенні

Функції введення та виведення

Специфікація форматів введення-виведення

Функція	Формат
scanf()	%[*] [ширина] тип
printf()	% [прапорець] [ширина] [.точність] тип

Специфікації деяких типів перетворення

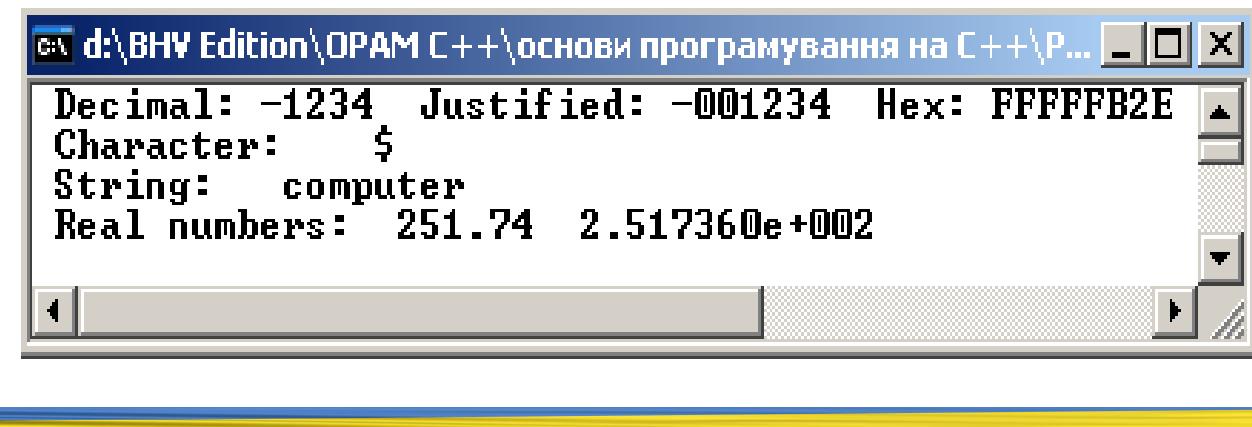
Символ формату	Тип аргументу
c	char
d	int (десятковий)
i	int (десятковий, вісімковий, шістнадцятковий)
D	long (десятковий)
o	int для цілого числа у вісімковій системі числення
O	long для цілого числа у вісімковій системі числення
x	int для цілого числа у шістнадцятковій системі числення
X	long для цілого числа у шістнадцятковій системі числення
f, e	float, double
s	Рядок символів
n	Покажчик на цілий тип
p	Покажчик на тип void (задає сегмент та зміщення)
+	Виведення знаків +/-, якщо використовується знаковий тип
-	Вирівнювання даних за лівим краєм

Приклад застосування функції printf()

Розглянемо функцію **printf()**, що виводить на екран:

- ціле від'ємне чотиризначне десяткове число,
- те саме число з точністю у шість розрядів
- те саме число в шістнадцятковій системі,
- символ з ширину поля у п'ять позицій,
- рядок символів,
- дійсне число у звичайній формі десяткового числа з точкою, подане у форматі **mmm.ddd**
- дійсне число в **експоненціальній формі**.

```
char ch = '$';
char *string = "computer";
int count = -1234;
double fp = 251.736;
printf(" Decimal: %d Justified: %.6d Hex: %X \n"
       " Character:%5c \n String: %10s\n"
       " Real numbers: %.2f %e \n",
       count,count,count,ch,string,fp,fp);
```



Приклад функцій введення-виведення



Приклад програми, в якій функції виведення використано для зображення на екрані значень у різних форматах.

[Код ex2_4.cpp](#)



Приклад функцій введення-виведення

```
//ex2_4.cpp          Демонстрація функцій введення-виведення
#include<stdio.h>    //бібліотека функцій введення та виведення
#include<conio.h>    //бібліотека консольного введення-виведення
#include<iostream>      //класи потоків введення-виведення
using namespace std;
void main()           //основна функція - точка входу до програми
{
    char simbol;           // символ, що вводиться
                           //вивести повідомлення на екран
    puts("input and show simbol using function _getch(),putchar()");
    putchar(_getch());     //вивести символ, що введений без луни
    puts("\n repeat input simbol without echo");
    simbol = _getch();
    char str[20];           //рядок, що вводиться
                           //вивести рядок
    puts("\n input string using function gets_s():");
    gets_s(str);           //ввести рядок
                           //функція форматованого виведення
    printf("output string using functions puts(): str=");
    puts(str);              //вивести рядок
    printf("\r\n output string using functions printf(): str=%s \r\n", str);
```



Приклад функцій введення-виведення

```
int i;  float a, b;          //числові дані, що вводяться
printf("\n input integer i, float a,b using function scanf_s()\n");
scanf_s("%d %f %e", &i, &a, &b);           //форматоване введення
printf("i=%d a=%f b=%e ", i, a, b);
cin.get();                  // читання коду ENTER
puts("\n input string for sscanf_s() as %d %f %f ");
char str1[100], str2[100];
fflush(stdin);              //очистити буфер потоку stdin
gets_s(str1);               //ввести рядок з клавіатури
printf("string is: ");
puts(str1);
//з рядка ввести дані за адресами операндів
sscanf_s(str1, "%d %f %f", &i, &a, &b);
printf("string is converting to i=%d a=%f b=%e \n", i, a, b);
puts("values from i,a,b are converting to string");
sprintf_s(str2, "i=%d a=%f b=%e ", i, a, b);      //вивести у рядок
puts(str2);
char string[30];
```



Приклад функцій введення-виведення

```
cout << "\n using cin, cout to input int, float, string values" << endl;
                                                // потокове виведення
cin >> i >> a >> string;                  // потокове введення
cout << " Data entry: i=" << i << " a=" << a << " string="" << string << "" << endl;
cin.get();                                     // читання коду ENTER
cout << "repeat input text" << endl;
cin.getline(string, 30);                      // потокове введення рядка символів
cout << " entry text = " << "" << string << "" << endl;
_getch();                                       // чекати натиснення клавіши
}
```



Сумісність типів у виразах

- ❑ Можливість присвоювання значень одного типу змінним іншого типу називають **сумісністю за присвоєнням**.
- ❑ Типи двох операндів у виразах можна вважати сумісними, якщо виконується хоча б одна з таких умов:
 - типи обох операндів **однакові**;
 - типи обох операндів **дійсні**;
 - типи обох операндів **цілочислові**;
 - тип одного операнду є **піддіапазоном** типу іншого операнду;
 - типи обох операндів є **піддіапазонами** одного й того самого базового типу;
 - тип одного операнду — **рядковий**, іншого — **рядковий або символний**.

УМОВИ СУМІСНОСТІ ЗА ПРИСВОЄННЯМ

Тип значення виразу є сумісним за присвоєнням з типом змінної, якій це значення надається, якщо виконується одна з таких умов:

тип змінної, яка набуває значення виразу, і тип виразу **однакові**;

тип змінної, яка набуває значення виразу, і тип виразу є **дійсними** типами, а діапазон значень типу виразу є піддіапазоном значень типу змінної;

тип виразу **ціличисловий**, а тип змінної, яка набуває значення виразу, **дійсний**;

тип виразу і тип змінної, яка набуває значення виразу, є **рядковими**;

тип виразу є **символьним**, а тип змінної, яка набуває значення виразу, — **рядковим**.

Приклад

Продемонструємо принцип використання операцій явного перетворення типів

```
d:\bhv edition\opam c++... Converting types Enter integer 65 <char>intvar=A Enter character h <int>symbolvar=104 <char>(intvar+5)=F <short>false=0
```

[Код ex2_5.cpp](#)

```
// ex2_5;
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
    int intvar;
    char symbolvar;
    cout<<"Converting types"=<<endl;
    cout<<"Enter integer"=<<endl;
    cin>>intvar;
    cout<<"(char)intvar="=<<(char)intvar<<endl;
    cout<<"Enter character"=<<endl;
    cin>>symbolvar;

    cout<<"(int)symbolvar="=<<(int)symbolvar<<endl;
    cout<<"(char)(intvar+5)="=<< (char)(intvar+5) <<endl;
    cout<<"(short>false="=<<(short>false<<endl;
    _getch();
}
```

Домашнє завдання

1. Обчислити значення виразів:

- а) $(2*2==4) \&\& \text{true};$
- б) $(2*2==4) \parallel \text{false};$
- в) $(!\text{true}) \parallel \text{false};$
- г) $(\text{true}==1) \wedge (\text{false}==0);$
- д) $2*\text{true}+3*\text{false};$
- е) $\text{false}==\text{true}==\text{false};$
- е) $58 \% 13 / 10;$
- ж) $9 \% 5 * 12 / 16.$

2. Обчислити значення виразу:

- а) $(\text{char})(\text{short})'0' + 9;$
- б) $(\text{char})(\text{short})'A' + 25;$
- в) $(\text{char})(\text{short})'0' - 16;$
- г) $'Z' > 'a';$
- д) $(\text{int})'9' - (\text{int})'0'.$

3. Записати вираз, який є істинним тоді й тільки тоді, коли дві прямі, що задані цілими коефіцієнтами рівнянь вигляду $ax + by + c = 0$:

- а) паралельні та не збігаються;
- б) паралельні (можливо, збігаються);
- в) збігаються;
- г) перетинаються;
- д) перпендикулярні.

4. Розставити дужки в таких виразах:

- а) $a=b+c*dd<<2\%8;$
- б) $a==b||a==c\&\&c<5;$
- в) $a=-b+++c---5;$
- г) $a=++b*2<<c\&dd+++3;$

Контрольні запитання

1. Що визначає тип даних?
2. Які стандартні типи даних є у мовах C/C++?
3. Що таке переповнення комірки оперативної пам'яті?
4. Охарактеризуйте прості типи.
5. Які операції означені для типів bool, int, float та char?
6. У чому різниця між типами char, int, long?
7. Що являють собою константи. Які є різновиди констант?
8. Що таке вираз?
9. Які вбудовані функції можна використовувати в константних виразах?
10. У чому полягає відмінність операцій / та %?
11. Який тип має результат операції відношення?
12. Яку структуру має програма, написана мовами C/C++?
13. Які дії виконує операція присвоєння?
14. У чому полягає відмінність між функціями потокового та консольного введення-виведення?
15. Які аргументи використовуються у функціях scanf_s() та printf()?
16. Що таке сумісність типів?



Дякую за увагу

Доц. кафедри ПСТ к.т.н. Ковалюк Т.В.

tkovalyuk@ukr.net

tetyana.Kovalyuk@knu.ua