Лекція 3.
Оператори в C++. Дебагер.
Ввід та вивід даних.

FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

План на сьогодні

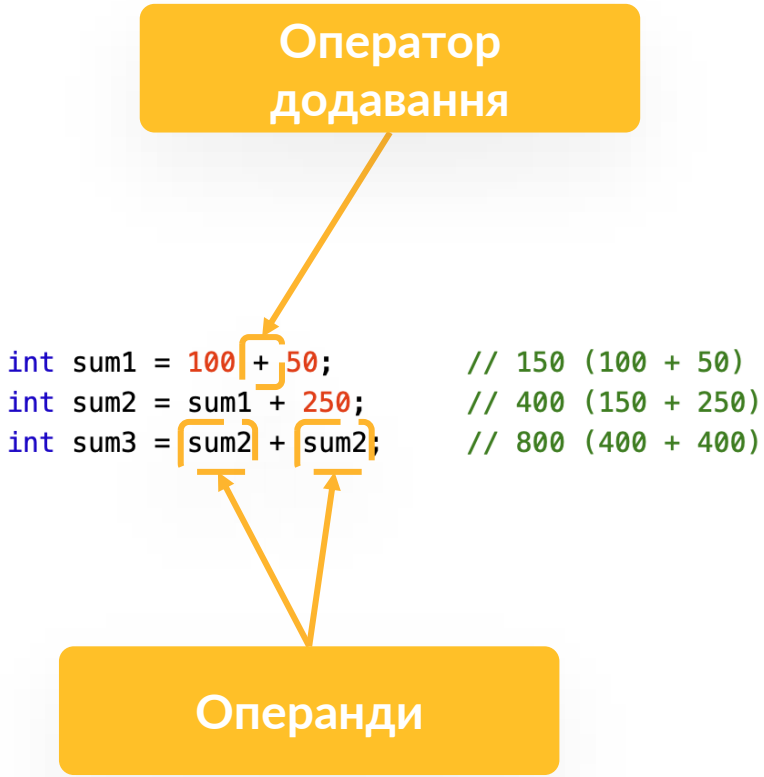
1. Що таке оператор?
2. Групи операторів
3. Класифікація операторів
4. Ввід та вивід даних
5. Коментарі
6. Виявлення помилок за допомогою дебагера

FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

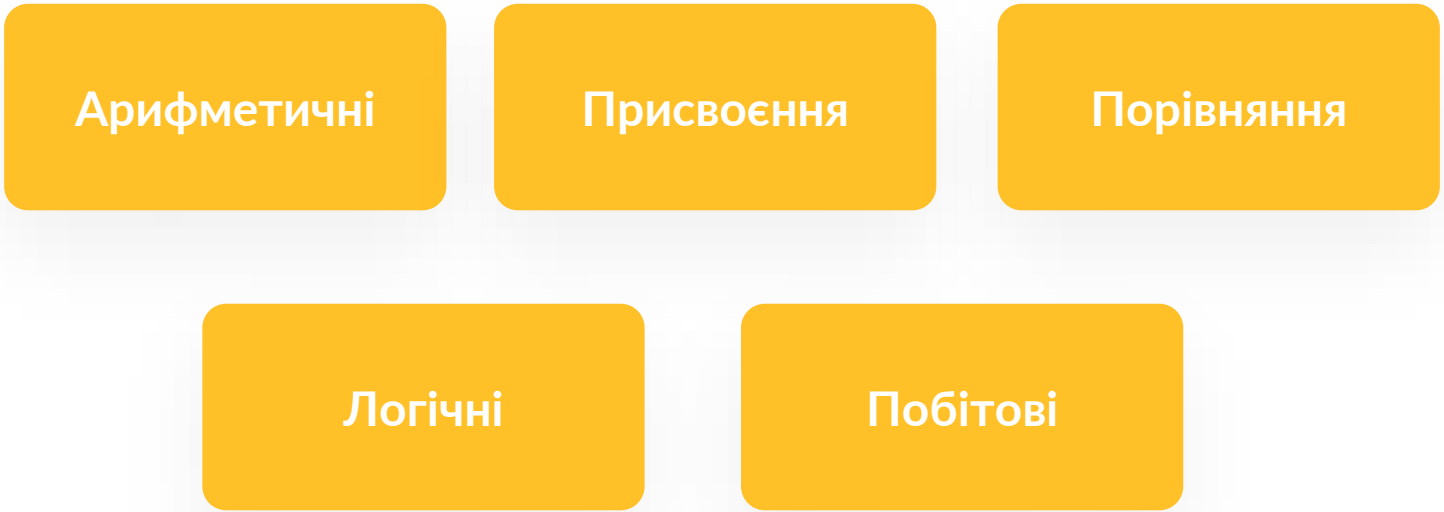
Що таке оператор?

Що таке оператор?

Оператор — це символ(и), які визначають операцію над змінними чи значеннями



Групи операторів в C++



Арифметичні

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

```cpp
#include <iostream>
using namespace std;

int main() {
  int x = 100 + 50;
  cout << x;
  return 0;
}
```

Оператор додавання

```
150
```

Присвоєння

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Присвоєння

```cpp
#include <iostream>
using namespace std;

int main() {
  int x = 10;
  x += 5; // теж саме, що x = x + 5;
  cout << x;
  return 0;
}
```

Додавання з присвоєнням

```
15
```

Порівняння

==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Порівняння

```cpp
#include <iostream>
using namespace std;

int main() {
  int x = 5;
  int y = 3;
  cout << (x > y); // returns 1 (true) because 5 is greater than 3
  return 0;
}
```

Оператор “більше”

```
1
```

Логічні

&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Логічні

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

int main() {
  int x = 5;
  int y = 3;
  cout << (x > 3 && x < 10); // returns true (1) because 5 is greater
than 3 AND 5 is less than 10
  return 0;
}
```

Логічне і

```
1
```

Побітові

Operator	Description
<<	Binary Left Shift Operator
>>	Binary Right Shift Operator
~	Binary One's Complement Operator
&	Binary AND Operator
^	Binary XOR Operator
|	Binary OR Operator

Побітове “і”

x	y	$x \,\&\, y$
0	0	0
0	1	0
1	0	0
1	1	1

```
#include <iostream>
#include <bitset>

using namespace std;

int main() {
  int a = 12, b = 25;

  cout << "a = " << a << endl;
  cout << "b = " << b << endl;
  cout << "a & b = " << (a & b) << endl << endl;

  cout << bitset<8>(a) << endl;
  cout << bitset<8>(b) << endl;
  cout << bitset<8>(a & b) << endl;
  return 0;
}
```

Оператор “побітове і”

```
a = 12
b = 25
a & b = 8

00001100
00011001
00001000
```

Побітове “або”

```cpp
#include <bitset>
#include <iostream>

using std::cout;
using std::endl;
using std::boolalpha;

int main() {
    int a = 12, b = 25;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "a | b = " << (a | b) << endl << endl;

    cout << std::bitset<8>(a) << endl;
    cout << std::bitset<8>(b) << endl;
    cout << std::bitset<8>(a | b) << endl;
    return 0;
}
```

Оператор “побітове або”

x	y	$x \mid y$
0	0	0
0	1	1
1	0	1
1	1	1

```
a = 12
b = 25
a | b = 29

00001100
00011001
00011101
```

Побітове xor

```cpp
#include <bitset>
#include <iostream>

using std::cout;
using std::endl;
using std::boolalpha;

int main() {
    int a = 12, b = 25;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "a ^ b = " << (a ^ b) << endl << endl;

    cout << std::bitset<8>(a) << endl;
    cout << std::bitset<8>(b) << endl;
    cout << std::bitset<8>(a ^ b) << endl;
    return 0;
}
```

Оператор “побітове xor”

x	y	$x \wedge y$
0	0	0
0	1	1
1	0	1
1	1	0

```
a = 12
b = 25
a ^ b = 21

00001100
00011001
00010101
```

Побітове xor

```
9   int main()
10  {
11      int a = 2, b = 3;
12
13      cout << "Before swapping a = " << a << " , b = " << b
14          << endl;
15      // applying algorithm
16      a ^= b; // a = a ^ b
17      b ^= a; // b = b ^ (a ^ b)
18      a ^= b; // a = (a ^ b) ^ a
19      cout << "After swapping a = " << a << " , b = " << b
20          << endl;
21      return 0;
22  }
```

Побітовий зсув

Оператор “побітовий зсув”

```
9   int main() {
10      {
11          int a = 12;
12          int b = a << 2;
13
14          cout << "a = " << a << endl;
15          cout << "b = " << b << endl << endl;
16
17          cout << std::bitset<8>(a) << endl;
18          cout << std::bitset<8>(b) << endl;
19      }
20      {
21          int a = 12;
22          int b = a >> 2;
23
24          cout << "a = " << a << endl;
25          cout << "b = " << b << endl << endl;
26
27          cout << std::bitset<8>(a) << endl;
28          cout << std::bitset<8>(b) << endl;
29      }
30      return 0;
31  }
```

```
a = 12
b = 48

00001100
00110000
a = 12
b = 3

00001100
00000011
```

Класифікація операторів

Унарні

1 операнд

Бінарні

2 операнди

Тернарний

3 операнди

Унарні (1 операнд)

Префіксні

++i;

--i;

&i;

*i;

+i;

-i;

Постфіксні

i++;

i--;

Різниця між i++ та ++i

```cpp
#include <iostream>
using namespace std;

int main() {
  int x = 10;
  int b = ++x; // перше додаємо, тоді присвоєюємо

  int y = 10;
  int c = y++; // перше присвоюємо, тоді додаємо

  cout << "x: " << x << " b: " << b << endl;
  cout << "y: " << y << " c: " << c << endl;

  return 0;
}
```

```
x: 11 b: 11
y: 11 c: 10
```

Бінарні (2 операнди)

a+b

a=b

a+=10

a<<i

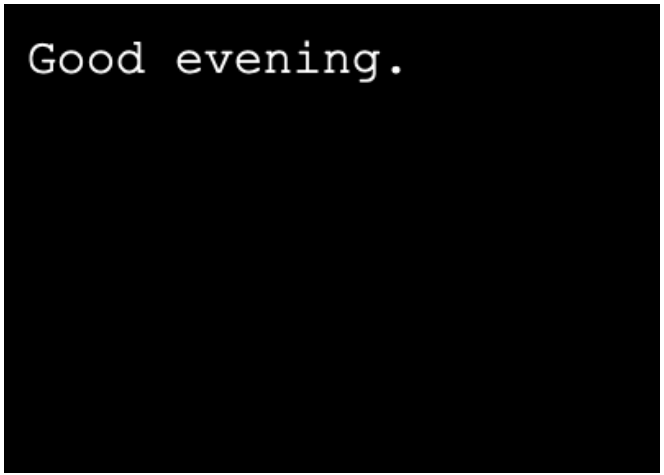
...

Тернарний оператор

Тернарний оператор

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

int main() {
  int time = 20;
  if (time < 18) {
    cout << "Good day.";
  } else {
    cout << "Good evening.";
  }
  return 0;
}
```

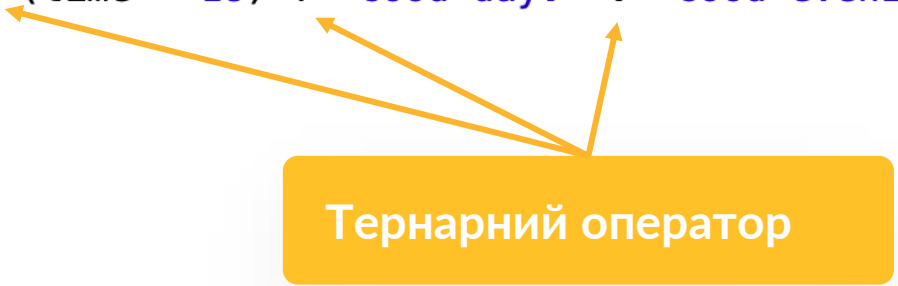


Тернарний оператор

```
Good evening.
```

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  int time = 20;
  string result = (time < 18) ? "Good day." : "Good evening.";
  cout << result;
  return 0;
}
```



Пріоритет операторів

Пріоритет операторів

Однаковий — для операцій з однієї групи пріоритет виконання за асоціативністю:

- **Правоасоціативні**: унарні та присвоєння
  a=b=c послідовність виконання a=(b=c)
- **Лівоасоціативні**: всі решта
  a+b+c послідовність виконання (a+b)+c
  -a*b-c послідовність виконання ((-a)*b) - c

Вищий — для операцій у дужках та в групах з меншим номером: 4+2*3 або (4+2)*3;

Пріоритет операторів

```cpp
#include<iostream>

using std::cout;
using std::endl;

int main()
{
    {
        int x = 8;
        int res = 4 * x++;
        std::cout << "res = " << res << ", x=" << x << endl;
    }
    {
        int x = 8;
        int res = 4 * --x;
        std::cout << "res = " << res << ", x=" << x << endl;
    }
    {
        int a = 1, b = 2, c = 5, k = 2;
        cout << a + b / c * (c -= k) << endl;
    }
    return 0;
}
```

Алгоритм виконання оператора

1. **Обчислення виразів в операндах**;
2. **Неявне перетворення типу операндів** у арифметичних і порівняльних діях, якщо він не підійшов точно. Мета – отримати такий тип операндів, який відповідає "найширшому" (допустимому у даному випадку) типові операндів;
3. **Виконання передбачуваної оператором дії над операндами**;
4. **Повернення результату** передбачуваного оператором типу.

Неявне перетворення типів

Умова	Перетворення
хоч один операнд `long double`	інший перетворюється до `long double`
якщо хоч один операнд `double`	інший перетворюється до `double`
якщо хоч один операнд `float`	інший перетворюється до `float`
інакше **для інтегральних типів**:	
якщо хоч один операнд `unsigned long`	інший перетворюється до `unsigned long`
якщо операнди `long` і `unsigned int`	обидва перетворюються до `long`
якщо один операнд `long`	то інший перетворюється до `long`
якщо один операнд `unsigned int`	то інший перетворюється `unsigned int`
інакше	обидва перетворюються до `int` (просування)

Неявне перетворення типів

```
7   int main()
8   {
9       {
10          short x = 4;
11          short y = 6;
12          cout << typeid(x + y).name() << " " << x + y << endl;
13      }
14      {
15          double x = 4.0;
16          int y = 6;
17          cout << typeid(x + y).name() << " " << x + y << endl;
18      }
19      {
20          int x = 1;
21          int y = 2;
22          cout << typeid(x/y).name() << " " << x/y << endl;
23      }
24      {
25          int x = 50000;
26          char y = x;     // переповнення типу
27          cout << static_cast<int>(y) << endl;
28      }
29
30      return 0;
31  }
```

Перетворення типів при присвоєнні

- При присвоєнні **нецілого числа змінній цілого типу**, дробова частина відкидається; якщо діапазон цілого типу замалий, то значення залишається невизначеним;
- При присвоєнні значення типу **bool змінній цілого типу**:

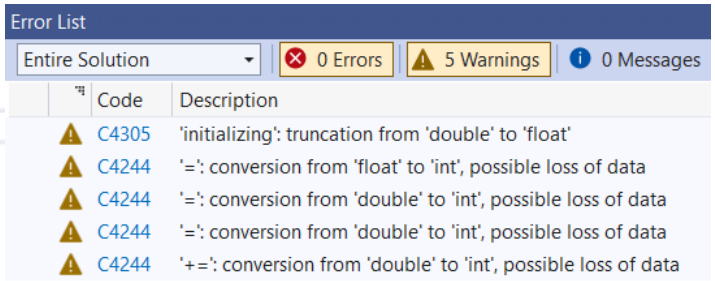
  true -> 1, false -> 0;

- При присвоєнні значення **цілого типу змінній типу bool**:

  0 -> false, ненульове значення -> true;
- При каскадному присвоєнні – дії справа наліво з узгодженням типу.

Перетворення типів при присвоєнні

```
1  #include <iostream>
2  #include <typeinfo>
3
4  using std::cout;
5  using std::endl;
6
7  int main()
8  {
9      int i_val;
10     float f_val = 4.6;
11     double d_val = 0.7e+10;
12     i_val = f_val;              // 4
13     cout << i_val << endl;
14     i_val = d_val;              // результат невизначений
15     cout << i_val << endl;
16     d_val = i_val = 3.541 + 3;  // 6
17     cout << d_val << endl;
18     d_val = 8.6;
19     i_val += d_val + 0.5;       //  15
20     cout << i_val;
21     return 0;
22 }
```



Особливості виконання дій у виразах

- **Вираз** – послідовність об'єктів і літералів, з'єднаних між собою операторами.
- **Побічні ефекти** – зміна під час обчислення виразу значень об'єктів, які входять у вираз.
- **Оптимізація виконання** логічних операторів && і || - правий операнд може не обчислюватись.
- **Порівняння на співпадіння (рівність)** float і double не має сенсу.

Особливості виконання дій у виразах

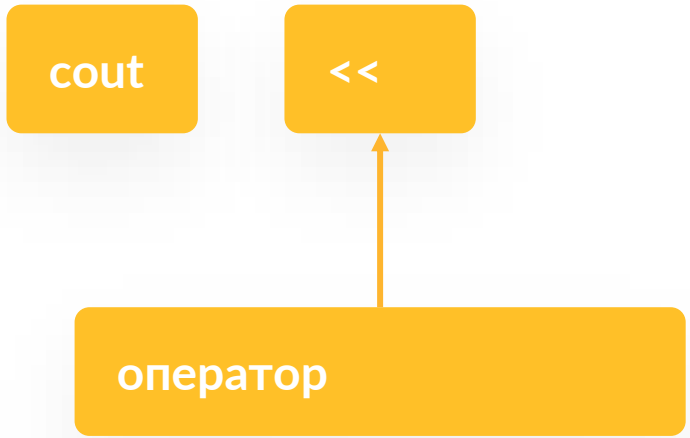
```
4  using std::cout;
5  using std::endl;
6  using std::boolalpha;
7
8  int main()
9  {
10     {
11         bool b_val = true;
12         int i_val = 1;
13         cout << boolalpha << (b_val && --i_val) << i_val << b_val + i_val << endl;
14     }
15     {
16         bool b_val = false;
17         int i_val = 1;
18         cout << boolalpha << (b_val && --i_val) << i_val << b_val + i_val << endl;
19     }
20     {
21         bool b_val = true;
22         int i_val = 0;
23         cout << boolalpha << (b_val || i_val++) << i_val << b_val + i_val << endl;
24     }
25
26     return 0;
27 }
```

```
false01
false11
true01
```

Ввід та вивід даних

FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

Вивід даних




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

int main() {
  cout << "Hello World!";
  return 0;
}
```

Новий рядок

\n

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

int main() {
  cout << "Hello World! \n";
  cout << "I am learning C++";
  return 0;
}
```

```
Hello World!
I am learning C++
```

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

int main() {
  cout << "Hello World!" << "\n\n";
  cout << "I am learning C++";
  return 0;
}
```

```
Hello World!

I am learning C++
```

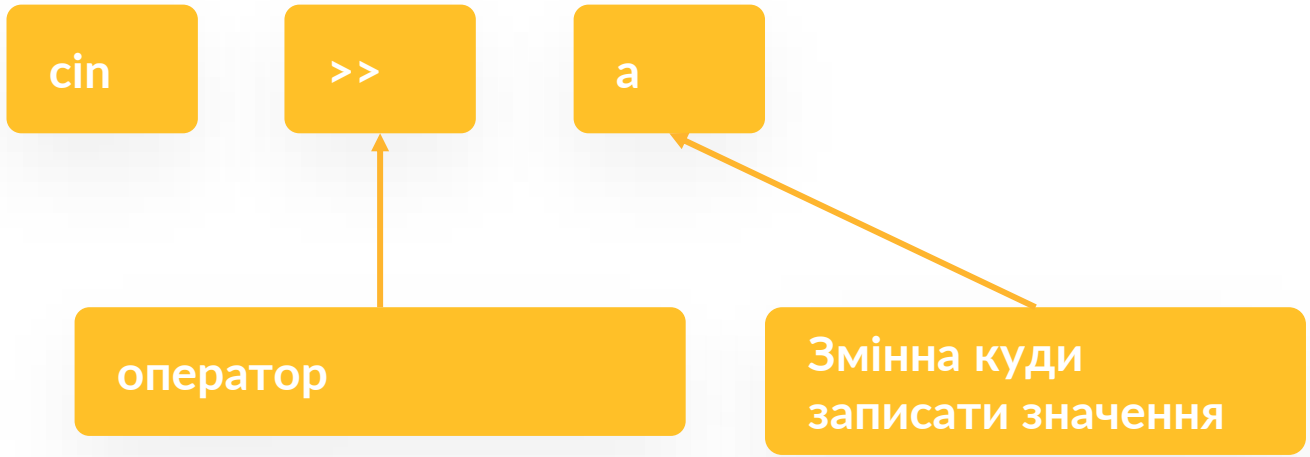
Новий рядок


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

int main() {
  cout << "Hello World!" << endl;
  cout << "I am learning C++";
  return 0;
}
```

```
Hello World!
I am learning C++
```

Ввід даних



Ввід даних

```
int x, y;
int sum;
cout << "Type a number: ";
cin >> x;
cout << "Type another number: ";
cin >> y;
sum = x + y;
cout << "Sum is: " << sum;
```

```
Type a number: 32
Type another number: 33
Sum is: 65
```

Коментарі

```cpp
#include <iostream>
using namespace std;

// код коментарів не виконується

int main() {
  int x = 10;
  int b = ++x;

  int y = 10;
  int c = y++;

  // коментується весь рядок
  // cout << "x: " << x << " b: " << b << endl;

  /*
  cout << "y: " << y << " c: " << c << endl;
  cout << "Hello, World" << endl;
  */

  return 0;
}
```

//

Коментує один рядок

/*....*/

Коментує текст від /* , і до */

FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

Дебагер

FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

Дебагер

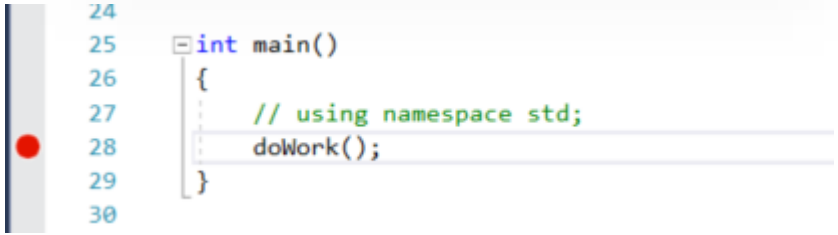
Дозволяє швидко виявляти проблеми в коді

Можна покроково виконувати код

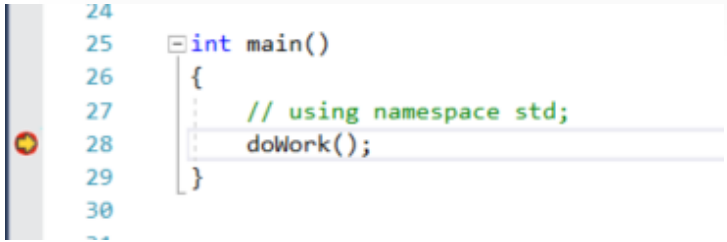
Можна спостерігати за значеннями змінних

Замість того, щоб писати багато cout <<”...”

F9 - поставити breakpoint



F5 - запустити програму з дебагером



Дякую!

FACULTY OF APPLIED
MATHEMATICS AND
INFORMATICS
LVIV UNIVERSITY