



# ПРИНЦИПИ СТВОРЕННЯ ТА ВИКОНАННЯ ПРОГРАМ

Лекція 2  
Основи інформатики, основи програмування та  
алгоритмічні мови

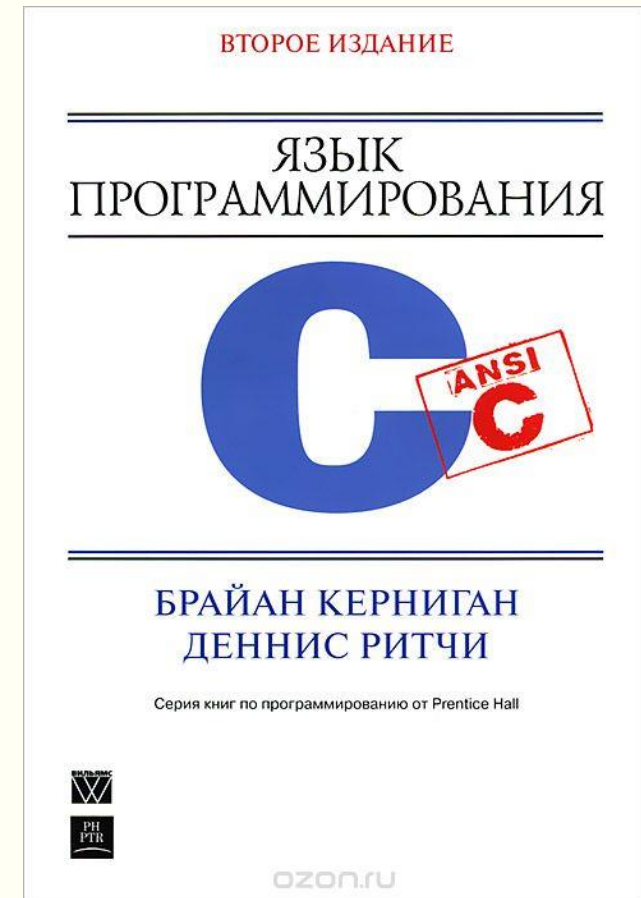
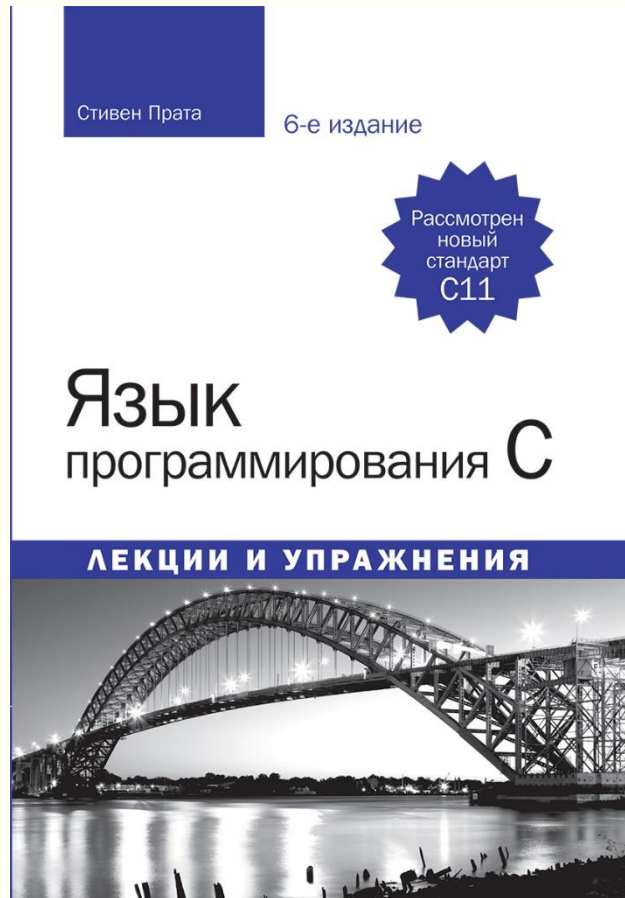


# План лекції

---

- Стандарти та структура програм мовою C
- Типізація мов програмування.
- Оператори мови програмування C.
- Операції вводу-виводу з символами та рядками.

# Рекомендована література (С)





# СТАНДАРТИ ТА СТРУКТУРА ПРОГРАМ МОВОЮ С

# Історія мови програмування С

---

**BCPL (1967, Мартін Річардс)** – мова для створення операційних систем та компіляторів

**В (1970, Кен Томпсон)** – використовувалась при створенні перших версій Unix у Bell Labs

**С (1972, Денніс Рітчі)** – мова розробки ОС Unix, практично незалежна від апаратної частини, для програмістів-практиків

- Кен Томпсон (зліва) і Денніс Рітчі (справа) за машиною PDP-11



# Стандарти мови програмування С

---

- Нині доступно багато реалізацій мови С.
  - В ідеалі, при написанні програми вона повинна працювати однаково з будь-якою з них, за умови, що не використовується специфічний для машини код.
  - Для цього вводять стандарти.
- Першим загальновизнаним стандартом став *класичний С (K&R С)*, що базувався на книзі «The C programming Language» Б. Кернігана та Д. Рітчі.
  - Проте в ньому не описано стандартну бібліотеку.
  - Фактичним стандартом стала стандартна бібліотека, що постачалась з реалізацією С для ОС Unix.
- По мірі розвитку та поширення мови виникла потреба у всезагальному, сучасному та строгому стандарті.
  - 1983р. – Національний інститут стандартизації США (ANSI) зібрав спеціальний комітет (X3J11) для його вироблення.



# Стандарти мови програмування C

---

- Перший офіційний стандарт ANSI C було прийнято в 1989 році.
  - Відповідний міжнародний стандарт ISO C погоджено в 1990 р.
  - Кінцеву версію стандарту називають C89/C90/ANSI C.
- Основні ідеї стандарту:
  - Довіряти програмісту.
  - Не заважати програмісту робити те, що він вважає необхідним.
  - Не збільшувати мову та зберігати її простоту.
  - Передбачати лише один спосіб виконання операції.
  - Робити операцію швидкодіючою, навіть якщо при цьому не гарантується переносимість.

# Стандарти мови програмування C

---

- Нова версія стандарту була прийнята в 1999 році (C99)
  - Мета: не додавати в мову нових рис, за виключенням необхідних
- Ключові доповнення (можуть не підтримуватись деякими компіляторами)
  - Додано підтримку інтернаціоналізації
  - Передбачено виправлення дефектів (при переході на x64)
  - Підвищено швидкодію мови для альтернативи FORTRAN
- Подальший розвиток стандарту – C11:
  - Пом'якшення цілі «довіри програмісту» з урахуванням безпеки та захисту
  - Деякі функціональні можливості C99 стали необов'язковими для C11
  - До необов'язкових можливостей була додана підтримка багатопоточності з огляду на поширення багатоядерних процесорів



# Переваги С

---

- **Ефективність:** мова продуктивно використовує комп'ютерні ресурси, володіє засобами точного управління, характерними для асемблера.
- **Переносимість:** написану на С програму з невеликими змінами можна виконати на іншій системі. Проте фрагменти коду, написаного для доступу до апаратних ресурсів зазвичай платформозалежні.
- **Потужність та гнучкість:** мовою С написані багато компіляторів та інтерпретаторів для інших мов (FORTRAN, Perl, Python, Pascal, LISP та ін.), інженерні розрахунки також часто виконуються на основі програм, написаних мовою С.
- **Орієнтація на програмістів:** мова надає доступ до обладнання та дозволяє маніпулювати окремими фрагментами пам'яті, має багатий вибір операцій для лаконічного вираження підходу до вирішення задач. Також представлений великий набір вбудованих бібліотек.

# Місце мови програмування C

---

## Галузі використання

- Операційні системи
- Компілятори та інтерпретатори мов програмування
- Текстові редактори
- Мережеві драйвери
- Сучасні програми та утиліти
- Бази даних

## C-подібні мови



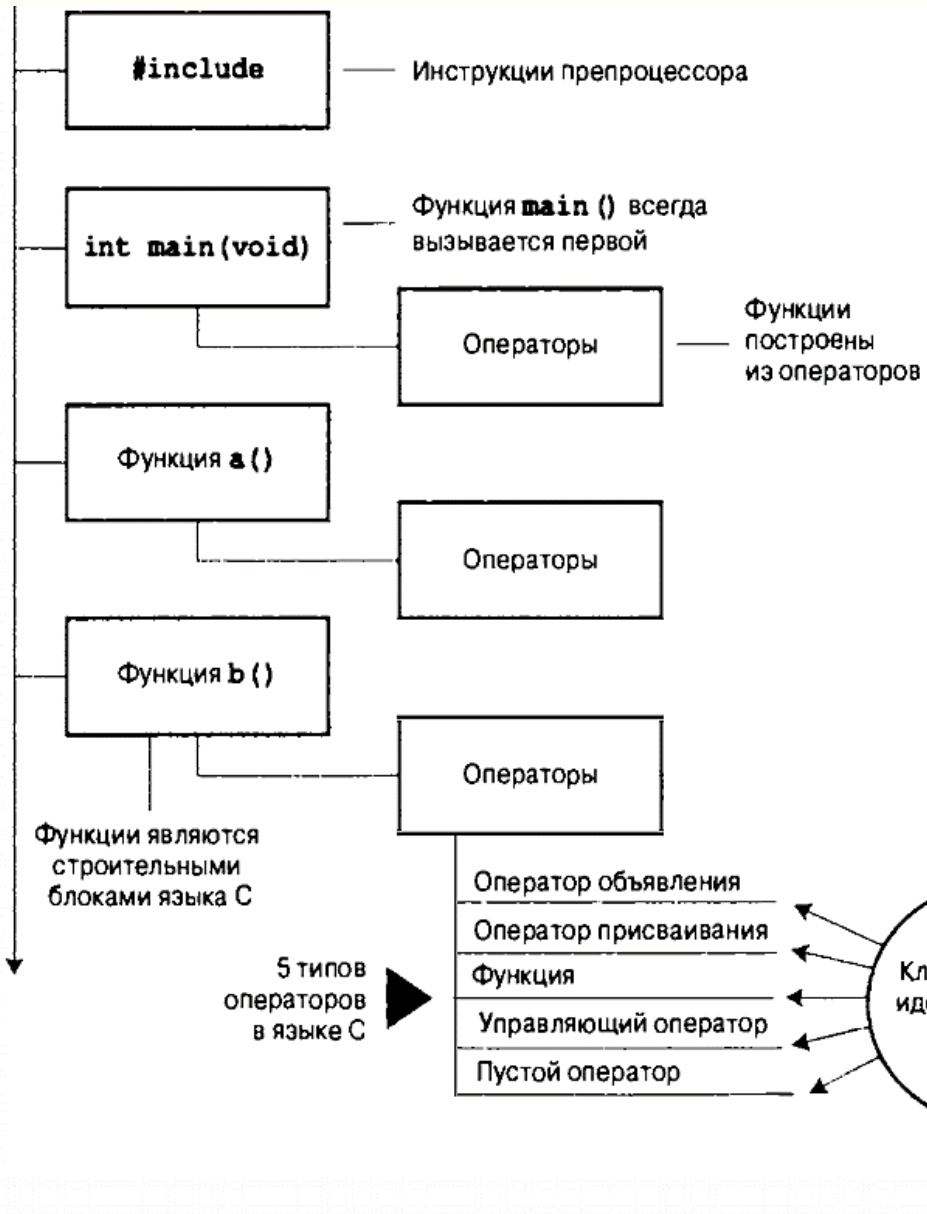
# Лістинг 1. Перша програма

---

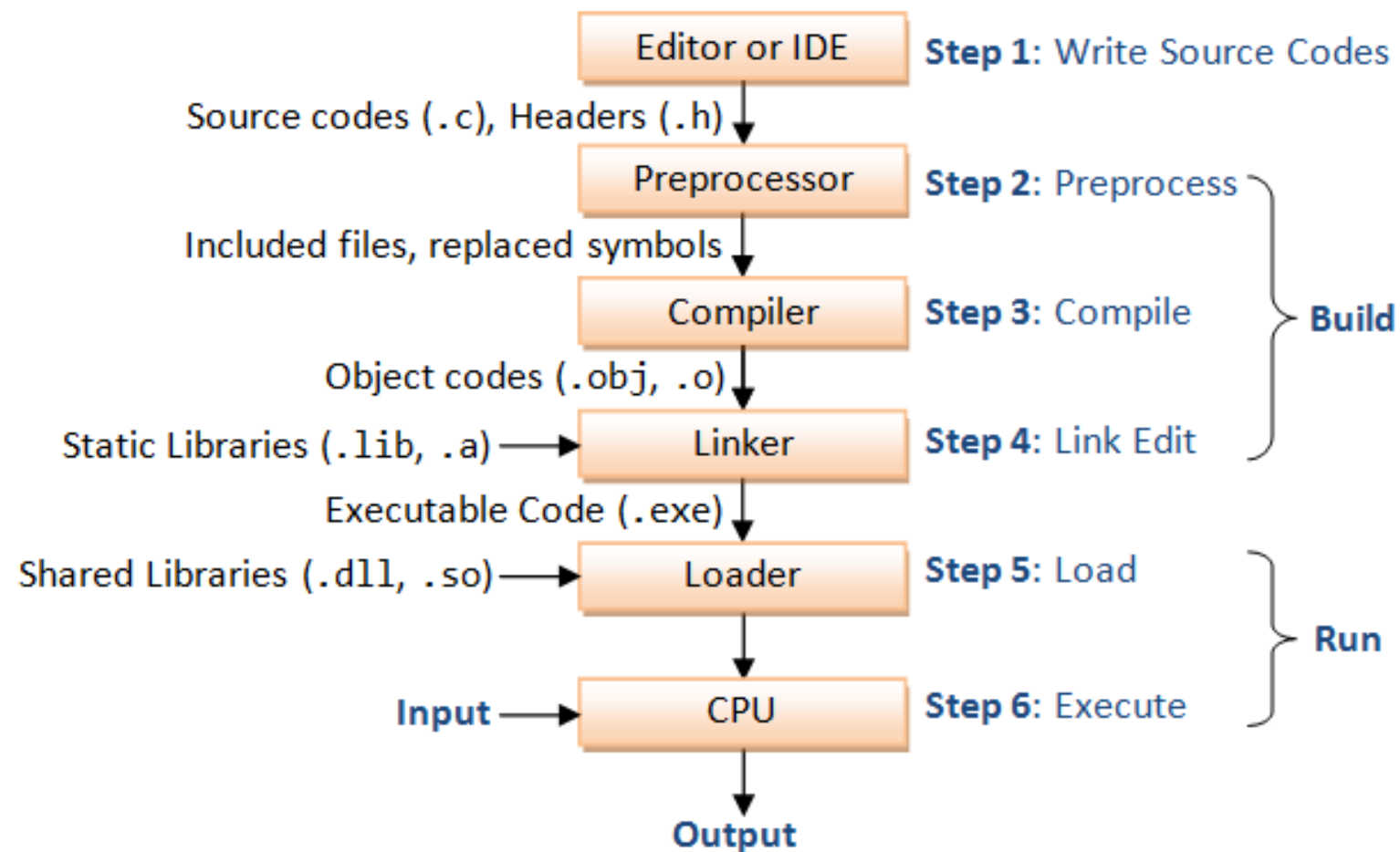
- Суть будь-якої програми зводиться до збереження, зміни та виводу деякої інформації.
  - Збереження – у змінних та константах
  - Зміна та вивід – у функціях

```
1  #include <stdio.h>
2  #include <locale.h>
3
4  int main(void)
5  {
6      // підтримка кирилиці
7      setlocale(LC_ALL, "");
8      int num; /* оголосити змінну з іменем num */
9      num = 1; /* присвоїти значення змінній num */
10     printf("Я простий ") ; /* використати функцію printf() */
11     printf("комп'ютер.\n");
12     printf ("Моєю улюбленою цифрою є %d, оскільки вона перша. \n", num);
13
14     return 0;
15 }
```

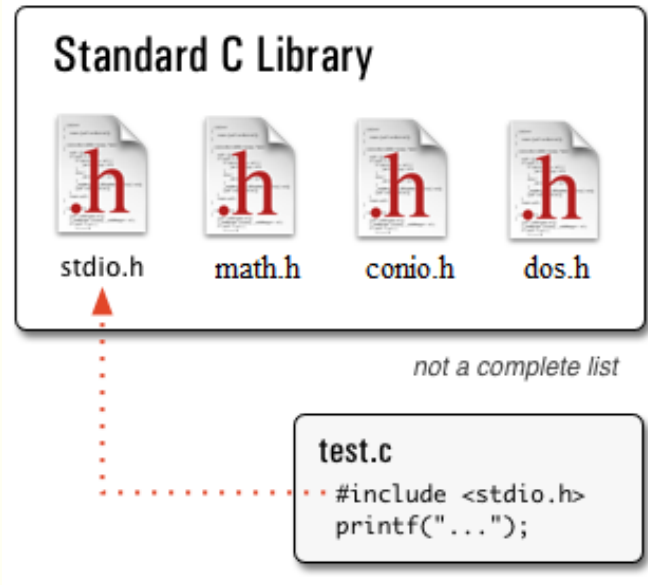
# Типова структура програми



```
1 #include <stdio.h>
2 #include <locale.h>
3
4 int main(void)
5 {
6     // підтримка кирилиці
7     setlocale(LC_ALL, "");
8     int num; /* оголосити змінну з іменем num */
9     num = 1; /* присвоїти значення змінній num */
10    printf("Я простий ") ; /* використати функцію printf() */
11    printf("комп'ютер.\n");
12    printf ("Моєю улюбленою цифрою є %d, оскільки вона перша. \n", num);
13
14    return 0;
15 }
```



# Деталі програми. Директива `#include` та заголовкові файли



- *Заголовкові файли* – файли з розширенням “.h”, що містять визначення групи функцій та змінних.
- `stdio.h` включає оголошення доступних у мові C функцій стандартного вводу-виводу: у програмі потрібно для виклику `printf()` і виводу інформації
- `locale.h` використовується для задач, пов’язаних з локалізацією: тут потрібен для виклику `setlocale()` та включення підтримки кирилиці

- **Корисні заголовкові файли:**

- `conio.h` – не є частиною C, корисний для консольного вводу-виводу (`getch()`, `kbhit()`)
- `stdlib.h` – функції для виділення пам’яті, перетворення типів тощо
- `math.h` – містить математичні функції

# Директиви препроцесора

---

- Препроцесор мови С являється макромовою, яка використовується для перетворення програми до того, як вона буде скомпільована.
  - Сама програма може бути написаною не тільки на С, але й С++, Objective-C або навіть асемблері.
- Директиви — це спеціальні команди, які препроцесор розпізнає та виконує.
  - Підключення файлів: `#include`, `#include_next` (перший знайдений файл).
  - Умовна компіляція: `#if`, `#ifdef`, `#ifndef`, `#else`, `#elif` and `#endif`.
  - Діагностика: `#error`, `#warning`, `#line`.
  - Додаткова інформація компілятору: `#pragma`
  - Макровизначення: `#define`



# Функція `main()` – точка входу в програму

---

- Функція – іменований блок коду, який можна повторно викликати та використовувати.
- Кожна функція повинна мати оголошення і визначення
  - *Заголовок (сигнатура, прототип) функції*: тут – `int main (void)`
  - *Фігурні дужки { та }*, що відокремлюють *тіло функції* (рядки 6-14)

```
4  int main(void)
5  {
6      // підтримка кирилиці
7      setlocale(LC_ALL, "");
8      int num; /* оголосити змінну з іменем num */
9      num = 1; /* присвоїти значення змінній num */
10     printf("Я простий ") ; /* використати функцію printf() */
11     printf("комп'ютер.\n");
12     printf ("Моєю улюбленою цифрою є %d, оскільки вона перша. \n", num);
13
14     return 0;
15 }
```

# Заголовки функцій

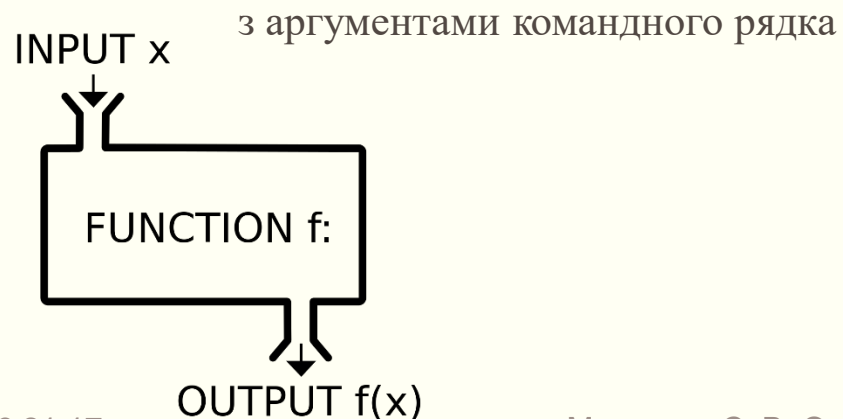
---

- Заголовок функції містить:

- Тип даних, що повертає функція (тут – `int`, пов'язаний з оператором `return`; функція `main()` повертає код помилки, що виникає в програмі).
- Назву функції (`main`)
- Список параметрів функції (тут – `void`)

- Можливе використання інших заголовків:

- `main()`
- `int main(void)`
- `int main(int argc, char *argv[])` –



```
Администратор: Командная строка
Microsoft Windows [Version 6.0.6002.18005]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Windows\system32>C:\readtxt2 read.txt
Could not open file read.txt

C:\Windows\system32>
```

# Тіло функції

---

- Містить набір інструкцій (операторів мови програмування, *statements*), що повинні виконуватись функцією.

- Інструкції змушують комп'ютер виконувати певні визначені дії.

- У мові C виділяють три види операторів мови програмування (інструкцій):

- **Інструкції-вирази (*expression statements*)**. Завжди закінчуються оператором ;

a = 6; c = a + b; ++j;

- **Складені інструкції (*compound statements*)**. Кілька інструкцій, що заключені в фігурні дужки. Не закінчуються оператором ;

{

    pi = 3.141593;

    circumference = 2. \* pi \* radius;

    area = pi \* radius \* radius;

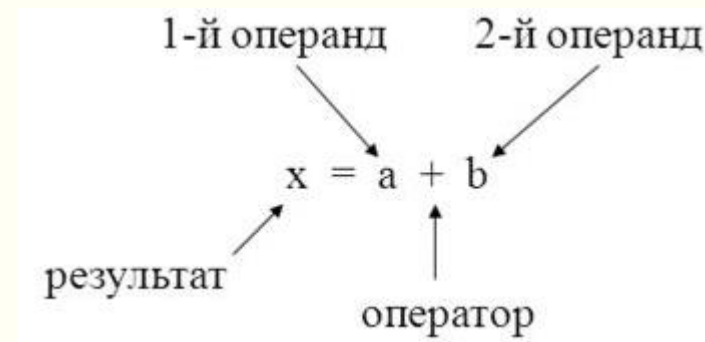
}

- **Управляючі інструкції (*control statements*)**. Визначають напрям виконання програми (галуження, цикли тощо)

# Інструкції-вирази (оператори виразу, expression statement )

---

- Вираз, що закінчується символом ;. Наприклад,
  - `i = 0;`
  - `i = i + 1;`
  - `printf("Hello, world!\n");`
  - `a = b*3 + c;`
- Найпростіша інструкція-вираз — пустий оператор ;
  - Він нічого не робить.
  - Проте використовується тоді, коли синтаксис вимагає присутності оператора виразу, проте він не потрібний.
  - Наприклад, у циклах `for`  
`for(int i = 1; i <= 10; i++)`  
`for( ; ; )`
- Вираз складається з операторів та операндів



# Вирази в мові С

Математичний вираз	Вираз С
$b^2 - 4ac = 0$	<code>b*b - 4*a*c==0</code>
$n < (a + b)^2$	<code>n&lt;(a+b)*(a+b)</code>
$\frac{x + y}{a} \geq \frac{x - y}{b}$	<code>(x+y)/a &gt;= (x-y)/b</code>
$\sqrt{x^2 + y^2} > ab$	<code>sqrt(x*x + y*y) &gt; a*b</code>

***Вирази мають (повертають) значення, яке отримується та записується за допомогою оператора присвоєння***

Оператор оголошення	<code>int toes;</code>
Оператор присвоєння	<code>toes = 12;</code>
Оператор виклику функції	<code>printf("%d\n", toes);</code>
Структурований оператор	<code>while (toes &lt; 20) toes = toes + 2;</code>
Оператор повернення	<code>return 0;</code>
Порожній оператор	<code>; /* нічого не робить */</code>

# Оператори (operators) у мові програмування C

---

## ■ *Арифметичні оператори:*

- присвоєння( $a=b$ ), додавання ( $a+b$ ), унарний мінус ( $-a$ ), остача від ділення ( $a\%b$ ), інкремент ( $a++$ ), декремент ( $a--$ ) та ін.

## ■ *Оператори порівняння:*

- рівність( $a==b$ ), нерівність ( $a!=b$ ), більше ( $a>b$ ), менше ( $a < b$ ), більше або дорівнює ( $a >= b$ ), менше або дорівнює ( $a <= b$ )

## ■ *Логічні оператори:*

- логічне заперечення (НЕ) ( $!a$ ), логічне І ( $a \&\& b$ ), логічне АБО ( $a \parallel b$ )

## ■ *Побітові оператори:*

- Побітове НЕ ( $\sim a$ ), побітове І ( $a \& b$ ), побітове АБО ( $a | b$ ), побітове виключаюче АБО ( $a \wedge b$ ), побітовий зсув вліво ( $a << b$ ), побітовий зсув вправо ( $a >> b$ )

## ■ *Складені оператори присвоєння:*

- Додавання ( $a += b$ ), віднімання ( $a -= b$ ), множення ( $a *= b$ ) та ділення ( $a /= b$ ) з присвоєнням, отримання остачі з присвоєнням ( $a \% = b$ ), побітове І ( $a \&= b$ ), побітове АБО ( $a |= b$ ), побітове виключаюче АБО ( $a \wedge = b$ ) з присвоєнням, побітовий зсув вліво ( $a <<= b$ ) та вправо ( $a >>= b$ ) з присвоєнням

# Арифметичні оператори

```
1 #include <stdio.h>
2 int main()
3 {
4     int a = 9, b = 4, c;
5
6     c = a+b;
7     printf("a+b = %d \n",c);
8
9     c = a-b;
10    printf("a-b = %d \n",c);
11
12    c = a*b;
13    printf("a*b = %d \n",c);
14
15    c=a/b;
16    printf("a/b = %d \n",c);
17
18    c=a%b;
19    printf("Remainder when a divided by b = %d \n",c);
20
21    return 0;
22 }
```

C:\Users\spuasson\Desktop\Untitled1.exe

```
a+b = 13
a-b = 5
a*b = 36
a/b = 2
Remainder when a divided by b = 1
```



arithmetic\_operators.c



# Оператори інкременту та декременту

---

```
#include <stdio.h>
int main()
{
    int a = 10, b = 100;
    float c = 10.5, d = 100.5;

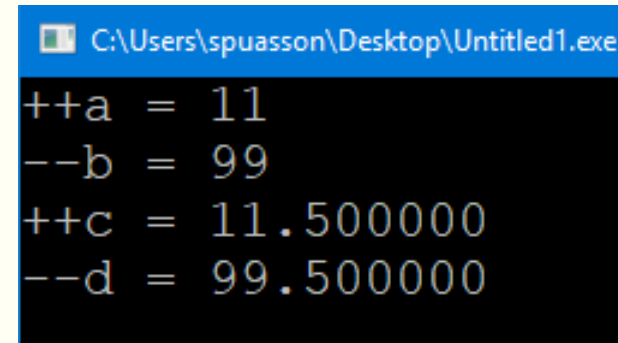
    printf("++a = %d \n", ++a);

    printf("--b = %d \n", --b);

    printf("++c = %f \n", ++c);

    printf("--d = %f \n", --d);

    return 0;
}
```



```
C:\Users\spuasson\Desktop\Untitled1.exe
++a = 11
--b = 99
++c = 11.500000
--d = 99.500000
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a = 5, c;
```

```
    c = a;
```

```
    printf("c = %d \n", c);
```

```
    c += a; // c = c+a
```

```
    printf("c = %d \n", c);
```

```
    c -= a; // c = c-a
```

```
    printf("c = %d \n", c);
```

```
    c *= a; // c = c*a
```

```
    printf("c = %d \n", c);
```

```
    c /= a; // c = c/a
```

```
    printf("c = %d \n", c);
```

```
    c %= a; // c = c%a
```

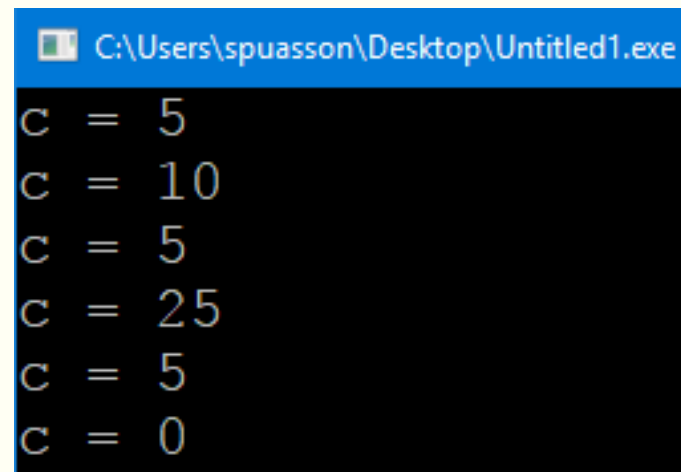
```
    printf("c = %d \n", c);
```

```
    return 0;
```

```
}
```

## Оператори присвоювання

---



```
C:\Users\spuasson\Desktop\Untitled1.exe
c = 5
c = 10
c = 5
c = 25
c = 5
c = 0
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a = 5, b = 5, c = 10;
```

```
    printf("%d == %d = %d \n", a, b, a == b); // true
```

```
    printf("%d == %d = %d \n", a, c, a == c); // false
```

```
    printf("%d > %d = %d \n", a, b, a > b); //false
```

```
    printf("%d > %d = %d \n", a, c, a > c); //false
```

```
    printf("%d < %d = %d \n", a, b, a < b); //false
```

```
    printf("%d < %d = %d \n", a, c, a < c); //true
```

```
    printf("%d != %d = %d \n", a, b, a != b); //false
```

```
    printf("%d != %d = %d \n", a, c, a != c); //true
```

```
    printf("%d >= %d = %d \n", a, b, a >= b); //true
```

```
    printf("%d >= %d = %d \n", a, c, a >= c); //false
```

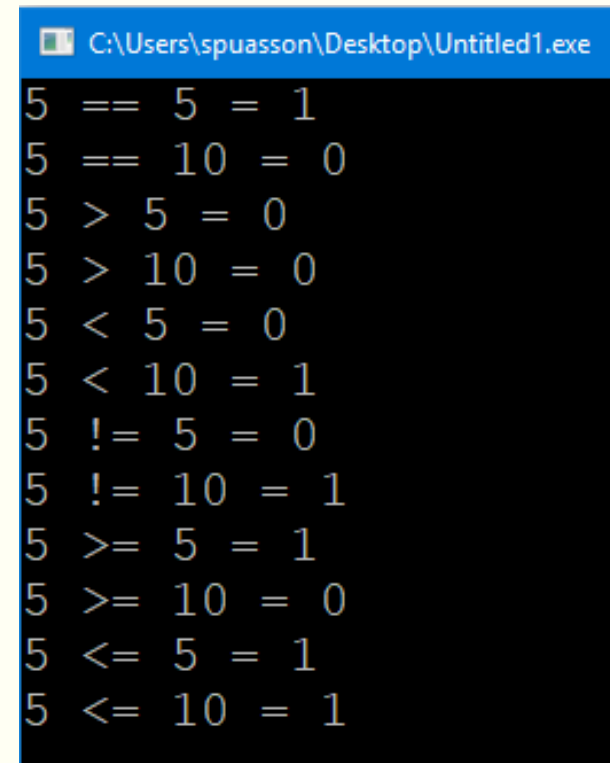
```
    printf("%d <= %d = %d \n", a, b, a <= b); //true
```

```
    printf("%d <= %d = %d \n", a, c, a <= c); //true
```

```
    return 0;
```

```
}
```

# Оператори порівняння



```
C:\Users\spuasson\Desktop\Untitled1.exe
5 == 5 = 1
5 == 10 = 0
5 > 5 = 0
5 > 10 = 0
5 < 5 = 0
5 < 10 = 1
5 != 5 = 0
5 != 10 = 1
5 >= 5 = 1
5 >= 10 = 0
5 <= 5 = 1
5 <= 10 = 1
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a = 5, b = 5, c = 10, result;
```

```
    result = (a == b) && (c > b);
```

```
    printf("(a == b) && (c > b) equals to %d \n", result);
```

```
    result = (a == b) && (c < b);
```

```
    printf("(a == b) && (c < b) equals to %d \n", result);
```

```
    result = (a == b) || (c < b);
```

```
    printf("(a == b) || (c < b) equals to %d \n", result);
```

```
    result = (a != b) || (c < b);
```

```
    printf("(a != b) || (c < b) equals to %d \n", result);
```

```
    result = !(a != b);
```

```
    printf("!(a == b) equals to %d \n", result);
```

```
    result = !(a == b);
```

```
    printf("!(a == b) equals to %d \n", result);
```

```
    return 0;
```

```
}
```

## Логічні оператори

C:\Users\spuasson\Desktop\Untitled1.exe

```
(a == b) && (c > b) equals to 1
(a == b) && (c < b) equals to 0
(a == b) || (c < b) equals to 1
(a != b) || (c < b) equals to 0
!(a == b) equals to 1
!(a == b) equals to 0
```

# Побітові оператори

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bit Operation of 12 and 25

00001100

& 00011001

00001000 = 8 (In decimal)

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise OR Operation of 12 and 25

00001100

| 00011001

00011101 = 29 (In decimal)

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise XOR Operation of 12 and 25

00001100

^ 00011001

00010101 = 21 (In decimal)

35 = 00100011 (In Binary)

Bitwise complement Operation of 35

~ 00100011

11011100 = 220 (In decimal)

212 = 11010100 (In binary)

212>>2 = 00110101 (In binary)

212>>7 = 00000001 (In binary)

212>>8 = 00000000

212>>0 = 11010100 (No Shift)

212 = 11010100 (In binary)

212<<1 = 110101000 (In binary) [Left shift by one bit]

212<<0 = 11010100 (Shift by 0)

212<<4 = 110101000000 (In binary) = 3392 (In decimal)

# Пріоритет операцій та категорії операторів

Приоритет	Оператор	Описание
1	++ --	Суффиксные/постфиксные инкремент и декремент
	()	Вызов функции
	[]	Обращение к элементу массива
	.	Обращение к члену структуры или объединения
	->	Обращение к члену структуры или объединения через указатель
	( type) { list }	Составной литерал(c99)
2	++ --	Префиксные инкремент и декремент
	+ -	Унарные плюс и минус
	! ~	Логическое НЕ и побитовое НЕ
	( type)	Приведение типа
	*	Разыменование
	&	Взятие адреса
	sizeof	Размер
	_Alignof	Выравнивание(c11)
3	* / %	Умножение, деление и остаток
4	+ -	Сложение и вычитание
5	<< >>	Побитовые левый сдвиг и правый сдвиг
6	< <=	Для операторов сравнения < и ≤ соответственно
	> >=	Для операторов сравнения > и ≥ соответственно
7	== !=	Для сравнений = и ≠ соответственно
8	&	Побитовое И
9	^	Побитовое XOR (исключающее или)
10		Побитовое ИЛИ (включающее или)
11	&&	Логическое И
12		Логическое ИЛИ
13	?:	Тернарное условие
14	=	Простое присваивание
	+= -=	Присваивание через сумму и разность
	*= /= %=	Присваивание через произведение, частное и остаток
	<<= >>=	Присваивание через левый сдвиг и правый сдвиг
	&= ^=  =	Присваивание через побитовые И, исключающее ИЛИ и ИЛИ
15	,	Запятая

Step 1.  $y = 2 * 5 * 5 + 3 * 5 + 7;$  (Leftmost multiplication)  
 $2 * 5$  is 10

Step 2.  $y = 10 * 5 + 3 * 5 + 7;$  (Leftmost multiplication)  
 $10 * 5$  is 50

Step 3.  $y = 50 + 3 * 5 + 7;$  (Multiplication before addition)  
 $3 * 5$  is 15

Step 4.  $y = 50 + 15 + 7;$  (Leftmost addition)  
 $50 + 15$  is 65

Step 5.  $y = 65 + 7;$  (Last addition)  
 $65 + 7$  is 72

Step 6.  $y = 72$  (Last operation—place 72 in y)

# Складені інструкції (compound statements, блоки)

---

- Група операторів МП, яку компілятор розглядає як єдиний оператор МП.
- Використовуються при написанні управляючих операторів, функцій

```
int add(int x, int y)
{ // початок блоку
    return x + y;
} // кінець блоку
```

- Блоки можна вкладати в інші блоки

```
void main()
{
    int num = 10;
    if(num > 0)
    {
        printf ("\Число додатне");
        printf ("\Це приклад складеного оператору");
    }
}
```



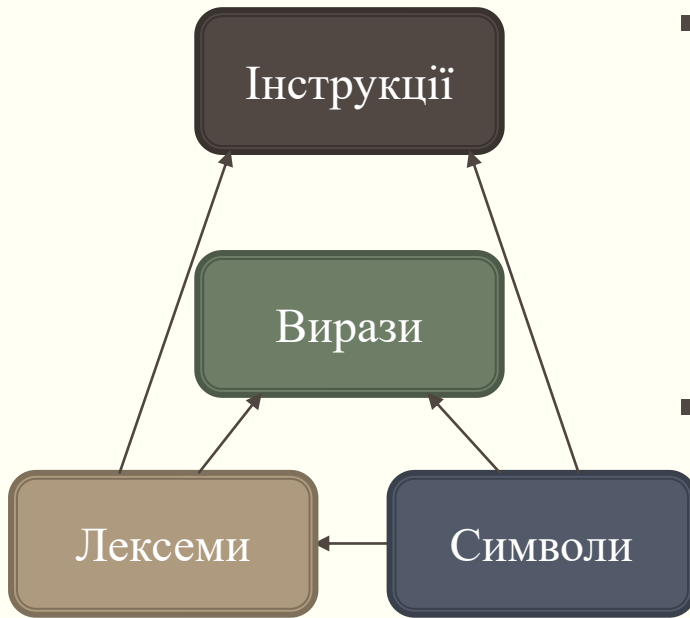
# Що виведе наступна програма?

---

```
main()
{
    int x;
    x = - 3 + 4 * 5 - 6; printf("%d\n",x);    (Операції 1.1)
    x = 3 + 4 % 5 - 6; printf("%d\n",x);      (Операції 1.2)
    x = - 3 * 4 % - 6 / 5; printf("%d\n",x);  (Операції 1.3)
    x = ( 7 + 6 ) % 5 / 2; printf("%d\n",x);  (Операції 1.4)
}
```

# Інструкції та вирази складаються з символів і лексем

---



- Мова С допускає використання символів:
  - латинського алфавіту
  - цифр
  - спеціальних символів: , (кома), ;, . (крапка), +, -, \*, ^, & (амперсанд), =, ~ (тільда), !, /, <, >, (, ), {, }, [, ], |, %, ?, ' (апостроф), " (лапки), : (двокрапка), \_ (знак підкреслення)
- **Лексема** – одиниця тексту програми, яка при компіляції сприймається як єдине ціле та за смислом не може бути розділена на менші елементи.
  - ідентифікатори
  - службові (ключові) слова
  - константи
  - рядки (рядкові константи)
  - операції (знаки операцій)
  - роздільники (знаки пунктуації)

# Ідентифікатори та ключові слова

- **Ідентифікатори** — це назви для змінних, типів, функцій та міток у тексті програми.
  - Написання та регістр символів у назвах ідентифікаторів повинні відрізнятись від усіх ключових слів.

auto	float	signed	<code>_Alignas</code> (начиная с C11)
break	for	sizeof	<code>_Alignof</code> (начиная с C11)
case	goto	static	<code>_Atomic</code> (начиная с C11)
char	if	struct	<code>_Bool</code> (начиная с C99)
const	<code>inline</code> (начиная с C99)	switch	<code>_Complex</code> (начиная с C99)
continue	int	typedef	<code>_Generic</code> (начиная с C11)
default	long	union	<code>_Imaginary</code> (начиная с C99)
do	register	unsigned	<code>_Noreturn</code> (начиная с C11)
double	<code>restrict</code> (начиная с C99)	void	<code>_Static_assert</code> (начиная с C11)
else	return	volatile	<code>_Thread_local</code> (начиная с C11)
enum	short	while	
extern			

```
#include <stdio.h>

int main() {
    int result;
    if ( result != 0 )
        printf( "Bad file handle\n" );
}
```

- Ідентифікатори можуть складатись як з одного, так і з кількох символів.
- Першим символом повинна бути буква або знак підкреслення (`_`), а за ним можуть іти букви, числа або знак підкреслення.

- **Ключові слова** — це слова, що використовуються для побудови мовних конструкцій.
  - Їх не можна використовувати для інших цілей.

# Угоди з нотації ідентифікаторів

## ▪ Верблюжа нотація (Camel Case)

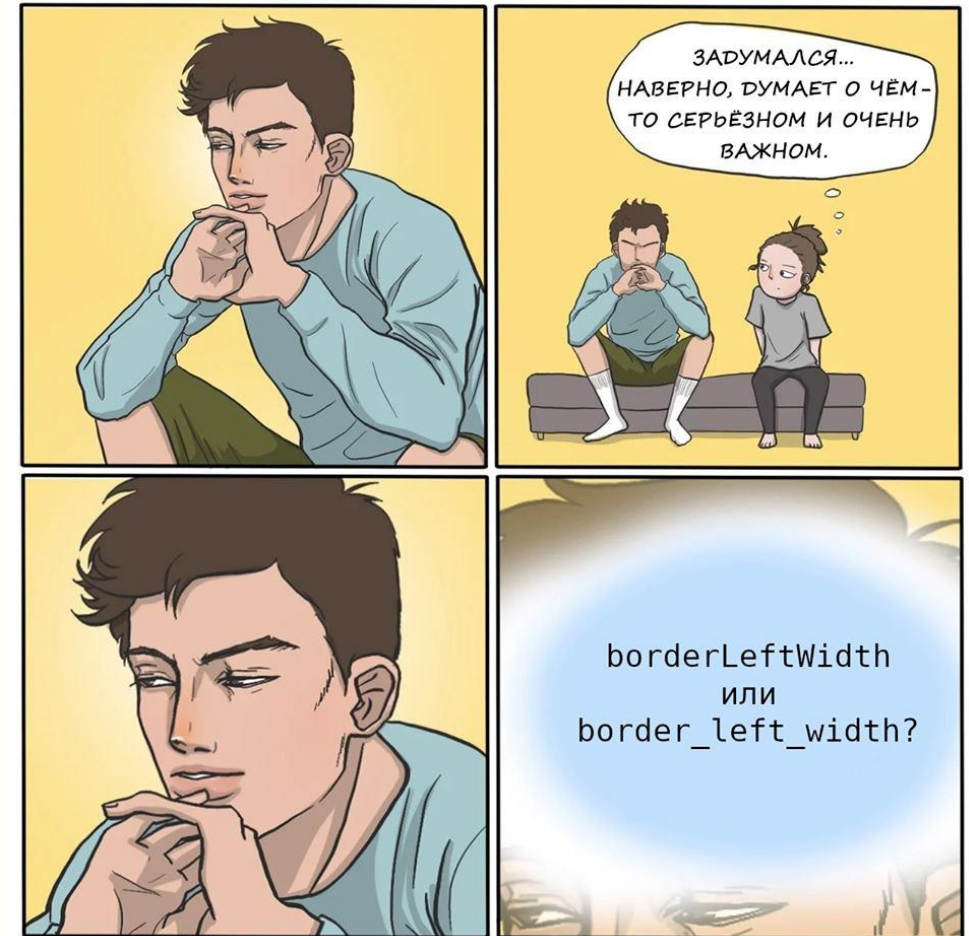
- стиль написання складових слів, у якому декілька слів пишуться разом без пробілів, при цьому *кожне слово пишеться з великої літери* (Pascal Case, верхній верблюдий регістр), *за винятком, можливо, першого* (нижній верблюдий регістр).
- Наприклад, UpperCamelCase (PascalCase), lowerCamelCase.

## ▪ Зміїна нотація (Snake Case, Underscore Case)

- стиль написання, у якому *слова розділяються не пробілами, а символом підкреслення* (\_), причому *кожне слово пишеться з малої літери*.
- Наприклад: str\_replace.

## ▪ Угорська нотація (Hungarian notation)

- метод найменування змінних в програмуванні, при якому *до ідентифікатора змінної або функції додається префікс, що вказує на його тип*.
- Наприклад, змінна sClientName – префікс s, що позначає тип змінної «рядок» (string).
- Була популярна в 90-х серед програмістів Microsoft



# Текст демонстраційної програми (маса в платиновому еквіваленті)

```
1  /* platinum.c  -- your weight in platinum */
2  #include <stdio.h>
3  int main(void)
4  {
5      float weight;    /* user weight          */
6      float value;     /* platinum equivalent */
7
8      printf("Are you worth your weight in platinum?\n");
9      printf("Let's check it out.\n");
10     printf("Please enter your weight in pounds: ");
11
12     /* get input from the user          */
13     scanf("%f", &weight);
14     /* assume platinum is $1700 per ounce */
15     /* 14.5833 converts pounds avd. to ounces troy */
16     value = 1700.0 * weight * 14.5833;
17     printf("Your weight in platinum is worth $%.2f.\n", value);
18     printf("You are easily worth that! If platinum prices drop,\n");
19     printf("eat more to maintain your value.\n");
20
21     return 0;
22 }
```

C:\Users\spuasson\AppData\Local\Temp\Rar\$Dla11328.16325\platinum.exe

```
Are you worth your weight in platinum?
Let's check it out.
Please enter your weight in pounds: 156
Your weight in platinum is worth $3867491.25.
You are easily worth that! If platinum prices drop,
eat more to maintain your value.
```



platinum.c

# Функції `scanf()` та `printf()` у роботі

```
/*platinum.c*/
```

```
int main(void)
```

```
{
```

```
scanf("%f", &weight)
```

Получение ввода с клавиатуры

```
printf("Ваш вес ...")
```

Отображение вывода программы

```
return 0;
```

```
}
```



# Стилі відступів (індентація)

---

Стиль K&R

```
while (x == y) {  
    something();  
    somethingelse();  
}
```

Стиль Олмана

```
while (x == y)  
{  
    something();  
    somethingelse();  
}
```

Стиль GNU

```
while (x == y)  
{  
    something();  
    somethingelse();  
}
```

Стиль Уайтсміта

```
while (x == y)  
{  
    something();  
    somethingelse();  
}
```





***Тема доповіді:*** препроцесор мови програмування C