Лекція 2

ОСНОВИ МОВИ СИСТЕМНОГО ПРОГРАМУВАННЯ С

Лекція 2. Основи мови системного програмування С

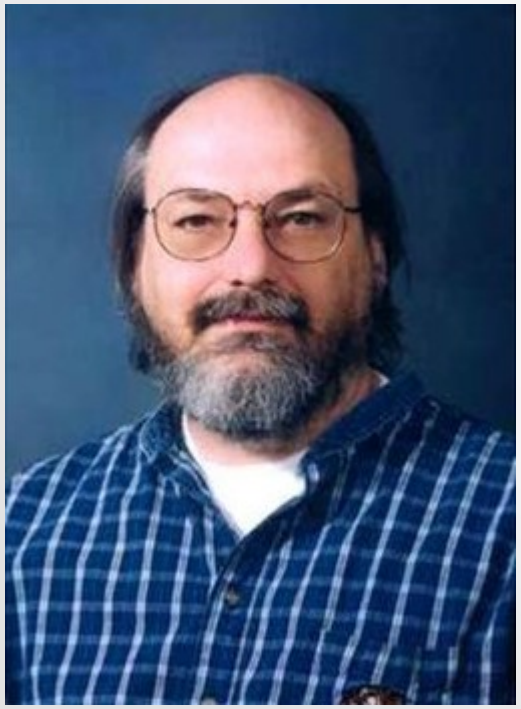
План

1. Історія створення мови програмування С
2. Типи даних, змінні, покажчики та константи в С
3. Структура програми мовою С
4. Передача параметрів у функції
5. Базові операції
6. Основні оператори
7. Ввід-вивід у С
8. Стандартна бібліотека

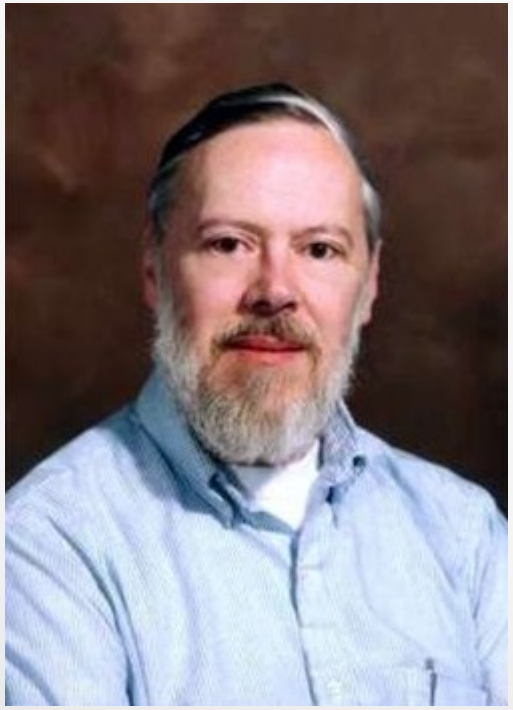
1. Історія створення мови програмування С

Мова програмування **С** була розроблена в 1969-1973 роках співробітником Bell Labs **Денісом Рітчі** як розвиток мови В, створеної раніше **Кеном Томпсоном**. Мова С спеціально створювалася для написання операційної системи **UNIX**.

Кен Томпсон



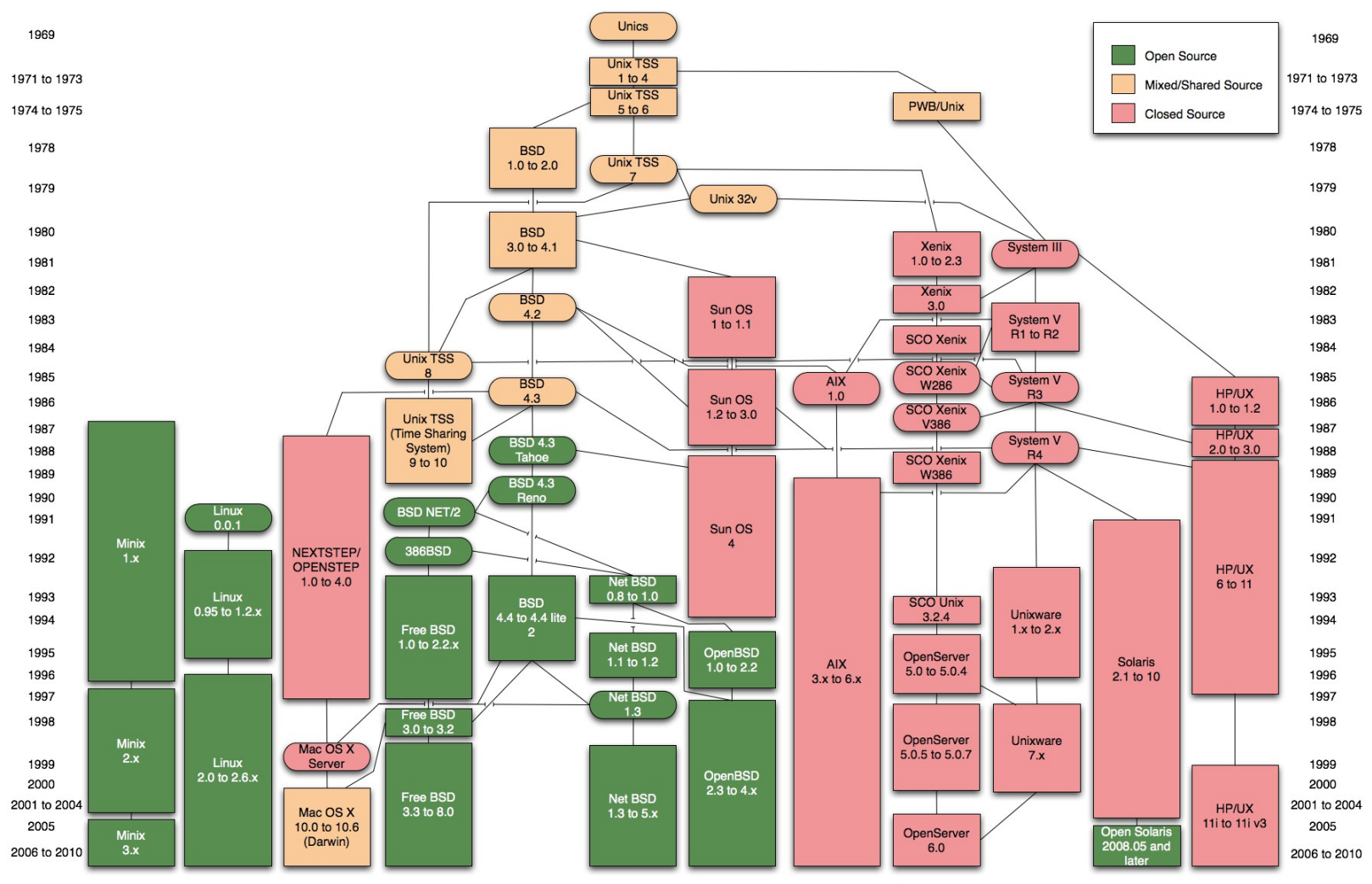
Деніс Ритчи



Б. Клінтон вручає К. Томпсону і Д. Рітчі Національну медаль в галузі технологій за “винахід операційної системи UNIX та мови програмування С”, 1999 р.

1. Історія створення мови програмування С

Генеалогічне дерево Unix-систем станом на 2010 рік



Мови на основі С:

- C++;
- C#;
- Objective-C;
- C--;
- D;
- Java;
- JavaScript;
- Perl;
- PHP;
- ...

2. Типи даних, змінні, покажчики та константи в С

Алфавіт мови С

Букви латинського алфавіту
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

Цифри
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Спеціальні символи
, (кома), ;, . (крапка), +, -, *, ^, & (амперсанд), =, ~ (тильда), !, /, <, >, (,), {, }, [,], |, %, ?, ' (апостроф), " (лапки), : (двокрапка), _ (знак підкреслення), \, #

З допустимих знаків формуються **лексеми** – визначені константи, ідентифікатори та знаки операцій. Лексеми є частиною виразів; а з виразів складаються інструкції та оператори мови С.

Важливо! Великі та малі літери в С вважаються різними!

2. Типи даних, змінні, покажчики та константи в С

Константи

Константи в мові С бувають цілими, дійсними і літералами.
Цілочисельні константи:

- 128 /* 128 в десятковій системі */
- 'A' /* ASCII-код символу 'A' */
- 0756 /* Число 494_{10} у вісімковій системі */
- 0xFF /* Число 255_{10} у шістнадцятковій системі */
- 0b1010101 /* Число 85_{10} у двійковій системі */

Дійсні константи:

- 3.14159 /* Константа у форматі double */
- 2.79F /* Константа у форматі float */

Літерали:

- "Hello world!"

2. Типи даних, змінні, покажчики та константи в С

Т**ип даних** у мові програмування – це множина допустимих значень та набір операцій, які можна застосовувати до цих значень.

В мові програмування С є лише вбудовані числові типи даних:
- цілі (**char**, **int**);
- дійсні (**float**, **double**).

Крім того, в С є спеціальний “пустий” тип (**void**), який використовується для оголошення функцій, що не повертають значення, або створення універсальних покажчиків.

Для вбудованих типів можуть використовуватись модифікатори:
- **signed** –значення зі знаком (за замовчуванням);
- **unsigned** – беззнакове значення;
- **short** – коротке;
- **long** – довге.

Наприклад, якщо тип даних **int** займає в пам’яті комп’ютера 2 байти, то за замовченням його значення змінюються від -2^{15} до $+2^{15}$, що відповідає неявному модифікатору **signed**. Типу даних **unsigned int** відповідає діапазон значень від 0 до 2^{16} -1. Тип даних **short int** буде в два рази коротшим типу **int** (тобто мати довжину 1 байт). Відповідно, тип **long int** буде в два рази довше **int** (4 байти).

2. Типи даних, змінні, покажчики та константи в С

Приклади оголошення змінних у мові С:

```
/* Приклад оголошення 4-х цілих змінних, одна з яких ініціалізована початковим значенням */
int a, A=4, b, my_variable;

// Оголошення та ініціалізація дійсних змінних
long double PI=3.14159, max_val=1.0E+10;
```

2. Типи даних, змінні, покажчики та константи в С

Крім вбудованих (атомарних) типів даних у С використовуються структуровані (складові) типи даних, до яких відносяться:

- **масиви** (одновимірні та багатовимірні) – індексовані набори однотипних елементів;
- **рядки** – одномірні масиви типу char з обов’язковим нульовим значенням наприкінці (нуль-терміновані рядки);
- **структури** (struct) – набори різних елементів (полів), що зберігаються як єдине ціле і що передбачають доступ до окремих полів;
- **об’єднання** (union) – спеціальний підвид структури, який реалізує можливість доступу до однієї й тієї ж області пам'яті, як до змінної, різних типів даних.

2. Типи даних, змінні, покажчики та константи в С

Приклади оголошення масивів у С:

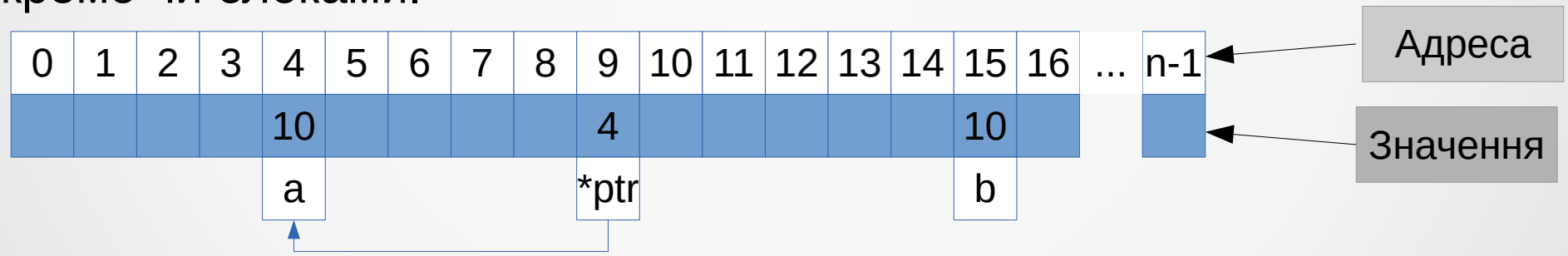
```
/* Приклади оголошень одновимірних масивів */
int array[5];
double data[] = {1.0, 2.0, 3.0, 4.0};

/* Приклади оголошення багатовимірних масивів */
int matr5x5[5][5],
    Matr2x3[][3] = {{1, 2, 3}, {4, 5, 6}};
```

Важливо! Індексація елементів масиву завжди починається з 0. Наприклад, елементами першого масиву на цьому плакаті будуть: array[0], array[1], …, array[4].

2. Типи даних, змінні, покажчики та константи в С

Покажчик – це спеціальний тип змінної, що містить адресу об’єкта (наприклад, змінної або функції). Покажчик не містить інформації про вміст об'єкта, а лише про його розміщення в пам’яті.

Пам’ять комп’ютера можна представити у вигляді послідовності пронумерованих однобайтових комірок, із якими можна працювати окремо чи блоками.




```
int a = 10, b; /* Оголошення цілих змінних */
int *ptr; /* Оголошення покажчика на змінну цілого типу */

ptr = &a; /* Присвоєння покажчику значення адреси змінної */
b = *ptr; /* Присвоєння змінній значення, на яке вказує покажчик */
```

2. Типи даних, змінні, покажчики та константи в С

Поняття покажчика був із масивами. Ім'я масиву є вказівником на його перший елемент. Звертатися до елементів масиву можна за їх індексами, а можна через покажчики. Наприклад:

```
int Array[5]; // Оголошення одновимірного масиву

Array[0] = 1; // Звернення до першого елементу масиву за індексом

* (Array + 4) = 10; // Звернення до останнього елементу масиву


int Matrix[5][5]; // Оголошення двовимірного масиву

*(*(Matrix + 3) + 3) = 4; // Matrix[3][3] = 4;
```

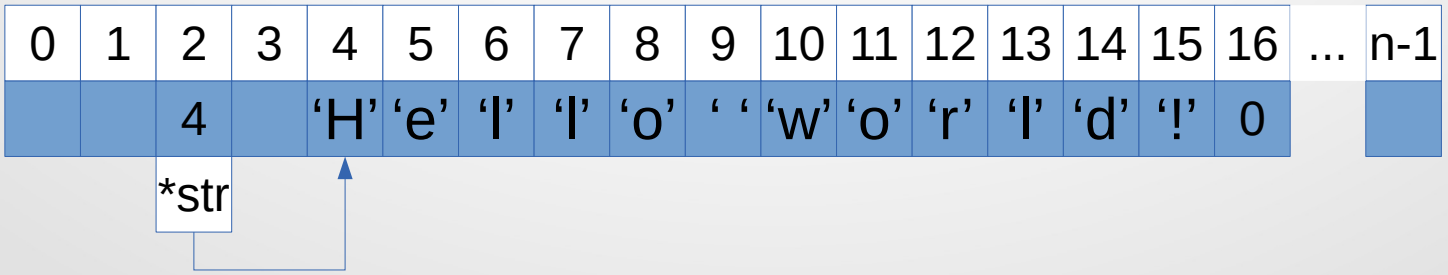
2. Типи даних, змінні, покажчики та константи в С

Рядок у мові С – це масив символів, наприкінці якого обов'язково перебуває спеціальна ознака кінця рядка NULL (символ із кодом ASCII – 0).

Рядок можна оголосити декількома способами.
1.
```
/* З використанням літералу */
char *str = "Hello world!";
```

2.
```
/* У явному вигляді як масив */
char hello[] = {'H', 'e', 'l', 'l', 'o', '', 'w', 'o', 'r', 'l', 'd', ' !', 0};
```

Схема розміщення рядка у пам’яті комп’ютера



2. Типи даних, змінні, покажчики та константи в С

Структури – це поєднання кількох об’єктів, можливо, різного типу під одним ім'ям, що є типом структури. В якості таких об'єктів (полів) можуть виступати змінні, масиви, покажчики та інші структури.

Наприклад:

```
/*  RGB - колір */
struct color
{
      int red;       /* Червона складова */
      int green;   /* Зелена */
      int blue;     /* Блакитна */
}

/* Приклад використання */
struct color white;

white.red = 255;
white.green = 255;
white.blue = 255;
```

3. Структура програми мовою С

Програма мовою С складається з однієї або більше підпрограм, які називаються **функціями**.

У загальному вигляді функція описується так:

[<тип_результату>]<ім'я_функції>([<список_аргументів>])
{
    [<тіло_функції>]
}

Наприклад, найпростіша програма мовою С може бути записана так:

main() {} // Деякі компілятори можуть не компілювати цей варіант

або так:

```
int main(void)
{
    return 0;
}
```

3. Структура програми мовою С

Точкою входу до програми є функція зі стандартним ім'ям main(). Зазвичай вона описується так:

```
int main(int argc, char* argv[])
{
    /* Оголошення змінних та функцій */
    /* … */
    /* Оператори, що реалізують алгоритм програми */
    /* … */
    return 0; /* Повернення операційній системі результату */
}
```

Параметр argc визначає кількість аргументів, що передаються в програму при її запуску з командного рядка, а argv є масивом покажчиків на рядки, що містять ці параметри. Причому рядок з ім'ям програми міститься за адресою argv[0] (параметр argc завжди більший за нуль).

3. Структура програми мовою С

У мові С будь-який об'єкт перед використанням повинен бути оголошений. Це означає, що якщо необхідно з однієї функції викликати іншу, то її опис (реалізація) або повинна розташовуватися вище за текстом програми, або в функції, що викликає, повинен бути описаний її шаблон.

Реалізація функції перед використанням

```
int sum(int a, int b)
{
	return a + b;
}

int main(void)
{
	int val1 = 10, val2 = 45;

	return sum(val1, val2);
}
```

Оголошення функції перед використанням

```
int main(void)
{
	int val1 = 10, val2 = 45;
	int sum(int, int); /* Оголошення функції */

	return sum(val1, val2);
}

int sum(int a, int b)
{
	return (a + b);
}
```

3. Структура програми мовою С

Шаблон (**прототип**, **сигнатура**) функції – це однозначний опис, що задає її ім'я, тип результату, що повертається, і кількість, а також типи прийнятих аргументів (параметрів). Імена аргументів у шаблоні можна опускати.

Наприклад:

```
/* Функція не приймає аргументів і не повертає жодного значення */
void sleep(void);

/* Функція приймає два покажчики цілого типу і нічого не повертає */
void swap (int * a, int * b);

/* Функція приймає змінну кількість аргументів і повертає ціле число */
int print(char *, ...);
```

3. Структура програми мовою С

Шаблони функцій стандартної бібліотеки мови С містяться у спеціальних заголовних файлах (що мають розширення *.h).
Наприклад, файл stdio.h містить заголовки стандартних функцій вводу-виводу, string.h – роботи з рядками і так далі.
Щоб не описувати стандартні (або бібліотечні) функції, заголовні файли можна підключати до вихідних текстів (стандартне розширення таких файлів – *.c) за допомогою спеціальної команди #include, яка є так званою **директивою препроцесора** мови С.

Наприклад,

```
#include <math.h> /* Підключення опису математичних функцій */

int main(void)
{
	float pi = 3.14159,
		val = sin (pi);

	/* … */
	return 0;
}
```

3. Структура програми мовою С

Препроцесор – це спеціальна програма, яка готує вихідний код мовою С до подальшої компіляції.
Препроцесор управляється так званими директивами, що починаються із символу #.

Найпопулярніші директиви:
#include – підключає вміст заданого файлу до поточного місця вихідного тексту програми;
#define - створення макропідстановки;
#if, #ifdef, #elif, #else, #endif - умовна компіляція.

Наприклад:

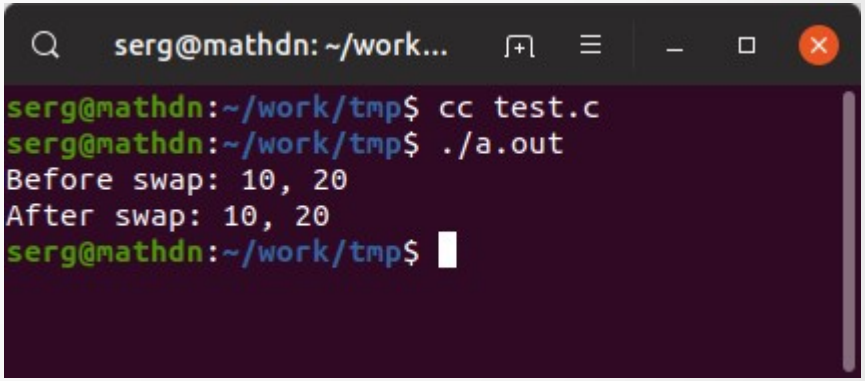
```
#include "myfile.h" /* Підключення файлу */

#define PI 3.14159 /* Створення константи */
#define SQR(x) (x)*(x) /* Створення макросу */

#define TED /* Умовна компіляція */
/* … */
#ifdef TED
	print("Hello Ted!");
#else
	print(“Hello everybody!”);
#endif
```

4. Передача параметрів у функції

Параметри функції у C передаються **за значенням**. Це означає, що при виході з функції значення аргументів не змінюється (точніше відновлюються оригінальні).

```
#include <stdio.h>
void swap(int a, int b)
{
   int tmp = a;

   a = b;
   b = tmp;
}

int main(void)
{
   int value1 = 10, value2 = 20;

   printf("Before swap: %d, %d\n", value1, value2);
   swap(value1, value2);
   printf("After swap: %d, %d\n", value1, value2);
   return 0;
}
```

Результат виконання

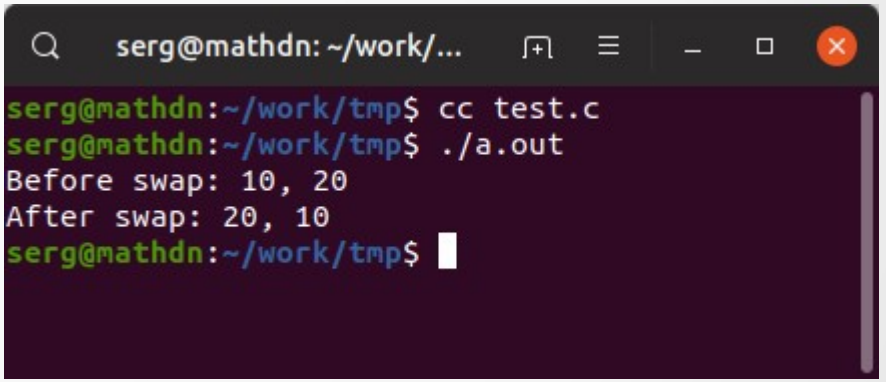


4. Передача параметрів у функції

Для зміни аргументів у тілі функції їх треба передавати у вигляді покажчиків.

```
#include <stdio.h>

void swap(int* a, int* b)
{
   int tmp = *a;

   *a = *b;
   *b = tmp;
}

int main(void)
{
   int value1 = 10, value2 = 20;

   printf("Before swap: %d, %d\n", value1, value2);
   swap(&value1, &value2);
   printf("After swap: %d, %d\n", value1, value2);
   return 0;
}
```

Результат виконання



5. Базові операції

У мові С над об’єктами можуть виконуватись такі основні операції:

- присвоювання;
- порівняння;
- арифметичні;
- логічні;
- побітові.

Результатом виконання операції є число.

Залежно від кількості операндів (аргументів) операція може бути **унарною**, **бінарною** чи **тернарною**.

5. Базові операції

Операція **присвоювання** позначається символом «=» і виконується у два етапи: 1) обчислюється вираз у правій частині; 2) результат присвоюється змінній, що стоїть у лівій частині.
Наприклад:

```
a = sin(3.14159) / 2 + cos(1);
```

Якщо об'єкти в лівій та правій частині операції присвоєння мають різні типи, використовується операція явного або неявного приведення (перетворення) типу.
Наприклад:

```
float val1 = 10.5;
int val2 = int(val1);  /* Явне приведення типу */
int val3 = val1;       /* Неявне приведення типу */
```

Важливо! Неявне перетворення типів є потенційно небезпечним!

5. Базові операції

Основні операції **порівняння** у мові С:

- == (порівняння);
- != (перевірка на нерівність);
- < (менше);
- > (більше);
- <= (менше чи дорівнює);
- >= (більше чи дорівнює).

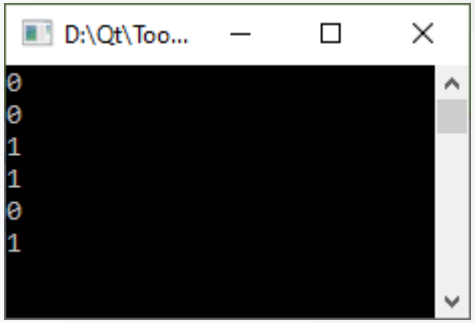
Операції порівняння застосовуються при організації умов і розгалужень. Результатом цих операцій є 0, якщо результат виконання операції не є істиною, та ненульове значення – навпаки.

5. Базові операції

Приклад роботи операцій порівняння

```
#include <stdio.h>

int main()
{
    int a = 3, b = 4, c;

    c = (a > b);
    printf("%d\n", c);
    c = (a >= b);
    printf("%d\n", c);
    c = (a < b);
    printf("%d\n", c);
    c = (a <= b);
    printf("%d\n", c);
    c = (a == b);
    printf("%d\n", c);
    c = (a != b);
    printf("%d\n", c);
    return 0;
}
```

Результат виконання програми



Для отримання складних логічних виразів використовуються так звані логічні зв’язки: «||» (або) та «&&» (і).
Наприклад:
(a < 1) && (b > 2) – кон’юнкція (логічне «і») виразів праворуч та ліворуч від знака кон’юнкції.

5. Базові операції

У мові С реалізовані побітові операції, основними серед яких є:

- & (кон'юнкція – бінарна операція, результат якої дорівнює 1 тільки коли обидва операнди дорівнюють 1);
- | (диз'юнкція – бінарна операція, результат якої дорівнює 1 коли хоча б один з операндів дорівнює 1);
- ~ (інверсія – унарна операція, результат якої дорівнює 0 якщо операнд дорівнює 1, і 1 – навпаки);
- ^ (виключна диз'юнкція – бінарна операція, результат якої дорівнює 1, якщо тільки один з двох операндів дорівнює 1).

Для кожного біта результат виконання операції буде отримано відповідно до наступної таблиці.

Результат виконання побітових операцій

a	b	a & b	a | b	~a	a ^ b
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

5. Базові операції

Основні **бінарні арифметичні операції**, розташовані у порядку зменшення пріоритету:

- * (множення);
- / (ділення);
- + (додавання);
- - (віднімання);
- % (залишок від цілочисельного ділення).

Основні **унарні операції**:

- ++ (інкремент – збільшення 1);
- -- (декремент – зменшення на 1);
- - (зміна знака).

5. Базові операції

Бінарні арифметичні операції можуть бути поєднані з операцією присвоєння:

- об’єкт * = вираз; /* об’єкт = об’єкт * вираз */
- об’єкт / = вираз; /* об’єкт = об’єкт / вираз */
- об’єкт + = вираз; /* об’єкт = об’єкт + вираз */
- об’єкт - = вираз; /* об'єкт = об’єкт - вираз */
- об’єкт % = вираз; /* об’єкт = об’єкт % вираз */

Наприклад:

```
b + = c; /* Еквівалентно b = b + c */
b -= c; /* Еквівалентно b = b - c */
b *= c; /* Еквівалентно b = b * c */
…
```

5. Базові операції

У мові С існує також єдина у своєму роді **тернарна операція**. Її синтаксис наступний:

<логічний вираз>? <вираз 1> : <вираз 2>

Ця операція повертає значення <вираз 1>, якщо <логічне вираження> істинно, і значення <вираз 2>, якщо навпаки.

Наприклад, операція привласнення може бути реалізована так:

```
c = a > b ? sin(3.14): cos(3.14);
```

6. Основні оператори

У мові С оператори можна розділити на три категорії:

- **присвоювання;**
- **розгалуження;**
- **циклічні.**

Умовні оператори мають кілька форм запису. Найпростішою є така:

```
if (<логічний вираз>)
    <оператор 1>;
[
else
    <оператор 2>;
]
```

```
float pi = 3.14159, fi = 0;

if (sin(fi) < fi && fi !=0)
{
    fi += 2 * cos(fi);
    /* … */
}
else
{
    fi -= 2 * cos(fi);
    /* … */
}
```

6. Основні оператори

Для зручності в С є оператор розгалуження (множинного вибору) switch. Його синтаксис наступний:

```
switch (<логічний_вираз>)
{
    case <константа_1>:
        <оператор_1>;
        [break;]
    case <константа_2>:
        <оператор_2>;
        [break;]
        /*. . .*/
    case <константа_n>:
        <оператор_n>;
        [break;]
    default:
        <оператор_за_замовченням>
}
```

```
/* Підрахунок кількості заданих символів
   у вхідному потоці */
char ch;
int counter_a = 0, counter_b = 0, counter_B = 0,
    counter_default = 0;

/* Запит символу */
/* ... */
switch (ch)
{
   /* Велику і малу літеру вважаємо однаковими */
   case 'a':
   case 'A':
      counter_a++;
      break;
   case 'b':
      counter_b++;
      break;
   case 'B':
      counter_B++;
      break;
   default :
      counter_default++;
}
```

6. Основні оператори

Оператори циклу в мові С бувають трьох видів:

- while (цикл із передумовою);
- do...while (цикл із постумовою);
- for (параметричний цикл із заданим числом повторень).

Цикл із передумовою має наступний синтаксис:

```
while (<логічна_умова>)
      <оператор>;
```

Наприклад:

```
float sum = 0;
int n = 1;

while (n < 100)
      sum += 1.0 / float(n++);
```

6. Основні оператори

Якщо <логічна_умова> не є спочатку істинною, цикл з передумовою не виконається жодного разу. Якщо ж потрібно, щоб тіло циклу хоч раз виконалося, то використовують **цикл із постумовою**:

```
do
    <оператор>;
while (<логічна_умова>);
```

Наприклад, наступний цикл виконається один раз:

```
float sum = 0;
int n = 1000;

do
    sum += 1.0 / float(n++);
while (n < 100)
```

6. Основні оператори

Параметричний цикл найуніверсальніший, його синтаксис наступний:

```
for (<ініціалізація>; <умова>; <приросщення>)
    <оператор>;
```

Наприклад:

```
int sum = 0;

for (int i = 0; i < 100; i++)
    sum += i;
```

7. Ввід-вивід у С

У мові С немає вбудованих операторів вводу-виводу, їх замінюють спеціальні функції стандартної бібліотеки, шаблони яких описані в заголовку stdio.h.

Для вводу-виводу найчастіше використовуються функції scanf() та printf(). Їхні шаблони мають вигляд:

```
int scanf(const char *format, …);
int printf(const char *format, …);
```

Ці функції приймають змінну кількість параметрів. Перший обов'язковий параметр – рядок, що визначає **формат вводу-виводу** інформації, і складається з керівних символів, безпосередньо тексту виведення та специфікаторів, які визначають формат вводу-виводу, довжину і т.п. аргументів. Наприклад, зчитування рядка та чисел може виглядати так:

```
…
int a;
float b;
char str[100];
…
scantf("%s %d %f", str, &a, &b);
…
```

7. Ввід-вивід у С

Виведення на екран текстового рядка та чисел може бути реалізовано так:

```
…
int a = 5;
float b = 10.5;
char *str = "This is a string";
…
printf("Text out: %s %d %f", str, a, b);
```

Основні специфікатори формату

Код	Значення
%c	Символ
%d	Десяткове число цілого типу зі знаком
%e	Наукова нотація (е нижнього регістру)
%E	Наукова нотація (Є верхнього регістру)
%f	Десяткове число з рухомою комою
%o	Вісімкове ціле число без знаку
%s	Рядок символів
%u	Десяткове число цілого типу без знака
%x	Шістнадцяткове ціле число без знака (літери нижнього регістру)
%X	Шістнадцяткове ціле число без знака (літери верхнього регістру)
%p	Покажчик

7. Ввід-вивід у С

Керівні символи в рядку формату вводу-виводу не виводяться на екран, а контролюють розташування символів, що вводяться або виводяться. Відмінною рисою керівного символу є наявність зворотного слеша перед ним.

До основних керівних символів відносяться:
- '\n' – перехід на наступний рядок;
- '\t' – горизонтальна табуляція;
- '\v' - вертикальна табуляція;
- '\b' – повернення на один символ;
- '\r' – повернення на початок рядка;
- '\a' – звуковий сигнал.

Для вводу-виводу в стандартній бібліотеці є ще досить багато різних функцій, наприклад, gets(), puts(), getchar() тощо.

8. Стандартна бібліотека

Стандартна бібліотека мови C (libc, crt) містить безліч функцій, що реалізують ввід-вивід, взаємодію з операційною системою, математичні обчислення, обробку рядків, роботу з пам’яттю тощо.

Шаблони функцій стандартної бібліотеки C описані у спеціальних заголовних файлах.

Наприклад:

- stdlib.h – базові функції вводу-виводу;
- stdlib.h – виділення пам'яті, управління процесами тощо;
- math.h – математичні функції;
- string.h – робота з рядками;
- complex.h – функції для роботи з комплексними числами;
- time.h – робота з календарними датами та часом;
- ...

8. Стандартна бібліотека

Порівняння двох рядків з використанням стандартної бібліотеки здійснюється наступним чином:

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

int main(void)
{
	char *key = "MyKey",
		buffer[100];

	puts("Enter key");
	gets(buffer);
	if (strcmp(buffer, key) != 0)
	{
		/* Рядки не збігаються */
		puts("Invalid key");
		return 1;
	}
	puts("Ok");
	return 0;
}
```

8. Стандартна бібліотека

Динамічне виділення пам’яті реалізується, наприклад, так:

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

/* Визначення середнього арифметичного заданого набору чисел */
int main(void)
{
	float avg = 0, *buffer = NULL;
	int size;

	puts("Enter size of array: ");
	scanf("%d", &size);
	if (size <= 0)
	{
		puts("Invalid size\n"); // Введено некоректний розмір
		return 1;
	}
	// Динамічне виділення необхідного обсягу пам'яті
	if ((buffer = (float*)(malloc(size * sizeof(int)))) == NULL)
	{
		puts("Error alocating memory!\n"); // Помилка виділення пам'яті
		return 1;
	}
	puts("Enter array: ");
	// Введення масиву та обчислення середнього арифметичного
	for (int i = 0; i < size; i++)
	{
		scanf("%f", &(buffer[i])); // Введення даних у динамічно виділену пам'ять
		avg += buffer[i];
	}
	printf("Avg: %f\n", avg / (float)size);
	free(buffer); // Звільнення пам'яті
	return 0;
}
```