

1. Основні поняття

Алгоритм (Algorithmi за араб. ім'ям математика аль-Хорезмі) — набір кінцевого числа вказівок, що описують порядок дій виконавця для досягнення певного результату.

Алгоритм характеризується наступними властивостями:

- 1) **Дискретність.** Алгоритм представляє процес рішення задач як порядок виконання деяких простих кроків.
- 2) **Визначеність.** Кожен крок алгоритму повинен бути точно визначений. Дії, які необхідно виконати, повинні бути строго і недвозначно визначені для кожного можливого випадку.
- 3) **Скінченність.** Алгоритм завжди повинен закінчуватися після виконання певного числа кроків. Кількість кроків може бути як завгодно великим.
- 4) **Універсальність.** Алгоритм повинен бути застосовний до різних наборів вихідних даних.
- 5) **Результативність.** Завершення алгоритму певними результатами.

Для опису алгоритмів використовуються такі форми запису:

- словесна;
- псевдокод;
- схематична.

Неодмінним атрибутом алгоритму є дані. Незалежно від змісту будь-які дані в пам'яті ЕОМ представляються послідовністю двійкових розрядів, а їх значеннями є відповідні бінарні числа.

Для опису реальних об'єктів використовуються структури даних. Під **структурою даних** в загальному випадку розуміють сукупність елементів даних і безліч зв'язків між ними.

Структури даних формуються за допомогою типів даних, посилань і операцій над ними.

Однак слід розрізняти поняття фізична і логічна структура даних.

Під **фізичною структурою даних** розуміють спосіб представлення даних в пам'яті ЕОМ. У той час як під **логічною структурою даних** розуміють реалізацію певного типу даних.

2. Типи даних

Класифікація типів даних

Тип даних – множина можливих значень, набір операцій, які можна застосовувати до таких значень, а також спосіб реалізації зберігання значень і виконання операцій.

Змінна – це область пам'яті, яка має ім'я в якій зберігається значення певного типу даних

Розрізняють прості, складові та інші типи даних.

Прості типи даних можна розділити на:

- цілочисельні,
- дійсні,
- символьні,
- перелічувані,
- логічні.

Складові (складні) типи даних можна розділити на:

- масиви,
- строки,
- структури,
- об'єднання,
- класи.

Також існують інші типи даних:

- покажчики,
- посилання,
- порожній тип.

Розглянемо більш детальніше деякі типи даних.

Перелічуваний тип

Перелічуваний тип – тип даних, безліч якого значень являє собою обмежений список ідентифікаторів.

Перелічуваний тип:

- полегшують написання і розуміння програм;
- дозволяє створити набір пов'язаних ідентифікаторів.

Перелічуваний тип найкраще використовувати при визначенні набору пов'язаних ідентифікаторів. Наприклад: палітра кольорів, список можливих помилок, дні тижня тощо.

Загальний синтаксис визначення шаблону перераховується типу має вигляд:

```
enum ім'я_шаблону{  
    ідентифікатор1,  
    ідентифікатори 2,  
    // інші ідентифікатори  
};
```

Приклад визначення шаблону перелічуваного типу днів тижня:

```
enum Days_of_week {
```

```
    Mon,
```

```
    Tue,
```

```
    Wed,
```

```
    Thu,
```

```
    Fri,
```

```
    Sat,
```

```
    Sun
```

```
};
```

Кожному ідентифікатору автоматично присвоюється цілочисельне значення в залежності від його позиції у списку перерахування. За замовчуванням, першому ідентифікатору присвоюється ціле число 0, а кожному наступному - на одиницю більше, ніж попередньому.

Замість перелічуваного типу можна було б використати оголошення констант наступним чином:

```
const char Mon = 0;
```

```
const char Tue = 1;
```

```
const char Wed = 2;
```

тощо.

Якщо необхідно, щоб перелічування відповідали певним значенням, наприклад, починалися з одиниці - константи перелічування необхідно ініціалізувати. У цьому випадку попередній приклад можна записати в наступному вигляді:

```
enum Days_of_week {  
    Mon=1,  
    Tue,  
    Wed,  
    Thu,  
    Fri,  
    Sat,  
    Sun  
};
```

В цьому випадку еквівалентно оголошенню констант наступним чином:

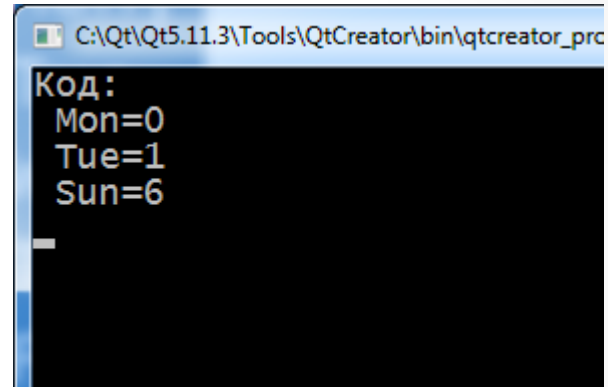
```
const char Mon = 1;  
const char Tue = 2;  
const char Wed = 3;  
тощо.
```

Приклад 1

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

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

```
int main()
{
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
    printf("Код:\n Mon=%d\n Tue=%d\n Sun=%d\n", Mon, Tue, Sun);
    return 0;
}
```

A screenshot of the Qt Creator application window. The title bar shows the path "C:\Qt\Qt5.11.3\Tools\QtCreator\bin\qtcreator_pr...". The main area is a black console window with the text "Код:" followed by "Mon=0", "Tue=1", and "Sun=6" on separate lines. The text is displayed in a light blue color on the black background.

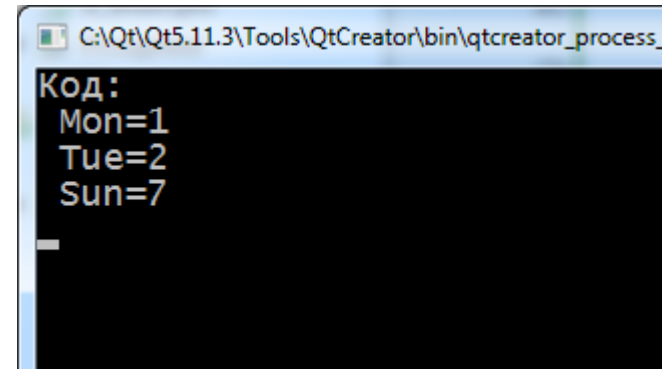
```
Код:
Mon=0
Tue=1
Sun=6
```


Приклад 2

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

enum Days_of_week {
    Mon=1,
    Tue,
    Wed,
    Thu,
    Fri,
    Sat,
    Sun
};

int main()
{
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
    printf("Код: \n Mon=%d \n Tue=%d \n Sun=%d \n", Mon, Tue, Sun);
    return 0;
}
```



The screenshot shows a console window titled "C:\Qt\Qt5.11.3\Tools\QtCreator\bin\qtcreator_process...". The output text is as follows:

```
Код:
Mon=1
Tue=2
Sun=7
```

Змінна перелічувального типу створюється та ініціалізується як звичайна змінна:

```
Days_of_week myday = Mon;
```

Хоча ідентифікатори цілочисельні, але ініціалізувати цілочисельними константами не можна!

```
Days_of_week myday = 5; ◦ cannot initialize a variable of ...
```

Для цього потрібно використовувати явне приведення типів:

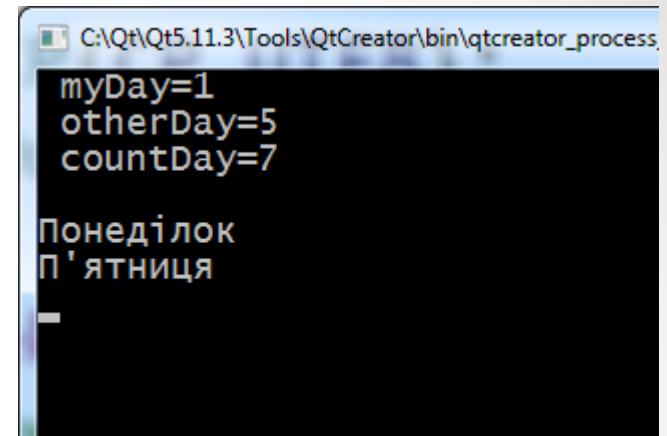
```
Days_of_week myday = (Days_of_week)5;
```

Користувацький тип можна використовувати в якості параметрів функції:

```
void printDayOfWeek(Days_of_week d) {  
    switch ((int)d) {  
        case Mon:  
            printf("Понеділок\n");  
            break;  
        case Tue:  
            printf("Вівторок\n");  
            break;  
        case Wed:  
            printf("Середа\n");  
            break;  
        case Thu:  
            printf("Четвер\n");  
            break;  
        case Fri:  
            printf("П'ятниця\n");  
            break;  
        case Sat:  
            printf("Субота\n");  
            break;  
        case Sun:  
            printf("Неділя\n");  
    }  
}
```

Приклад 3

```
• • •  
  
int main()  
{  
    SetConsoleCP(CP_UTF8);  
    SetConsoleOutputCP(CP_UTF8);  
  
    Days_of_week myDay = Mon;  
    //Days_of_week otherDay = 5;    // Ошибка  
    Days_of_week otherDay = (Days_of_week)5;  
    int countDay = Sun;  
    printf(" myDay=%d\n otherDay=%d\n countDay=%d\n\n", myDay, otherDay,  
countDay);  
  
    printDayOfWeek(myDay);  
    printDayOfWeek(otherDay);  
    //printDayOfWeek(countDay);    // Ошибка  
    //printDayOfWeek(25);    // Ошибка  
    return 0;  
}
```



Використання користувачького типу перелічення дозволяє виключити деякі помилки:

<u>printDayOfWeek</u> (countDay);	// Ошибка	◦ no matching function...
<u>printDayOfWeek</u> (25);	// Ошибка	◦ no matching function...

Структури

Структури – набір різних типів даних, що зберігаються як єдине ціле і що передбачають доступ до окремих полів структури. Для доступу до полів структури використовується точкова нотація.

Структури:

- полегшують написання і розуміння програм;
- допомагають згрупувати дані, що об'єднуються під загальним поняттям;
- дозволяють групу пов'язаних між собою змінних використовувати як множина окремих елементів, а також як єдине ціле.

Структури доцільно використовувати там, де необхідно об'єднати дані, що відносяться до одного об'єкту.

Загальний синтаксис визначення шаблону структури має вигляд:

```
struct ім'я_шаблону {  
    тип1 назва_змінної1;  
    тип2 назва_змінної2;  
    // інші ідентифікатори даних;  
};
```

Приклад визначення шаблону структури типу date:

```
struct date {  
    unsigned short nWeekDay;  
    unsigned short nMonthDay;  
    unsigned short nMonth;  
    unsigned short nYear; // останні 2 цифри року  
};
```

Звернення до певного члена структури проводиться за допомогою конструкції наступного виду:

`<ім'я змінної типу структура>.<ім'я елемента>`

Поля структур можуть мати довільний тип, як простий, так і складений: масив, символьний рядок або навіть вкладену структуру.

Для вкладених структур діють такі самі правила оголошення, як і для звичайних структур.

Приклад 4

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

```
enum Days_of_week {
    Mon=1,
    Tue,
    Wed,
    Thu,
    Fri,
    Sat,
    Sun
};
```

```
struct schedule {
    Days_of_week d;
    char lesson1[100]="";
    char lesson2[100]="";
    char lesson3[100]="";
    char lesson4[100]="";
    char lesson5[100]="";
    char lesson6[100]="";
};
```

```
int main()
{
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
    schedule day1 = {Mon, "OOP", "MMDO",
"Algorithms", "", "", ""};
    schedule day2;
    day2.d = Tue;
    sprintf(day2.lesson1, "%s", "OOP");
    sprintf(day2.lesson2, "%s",
"Algorithms");
    schedule day3;

    printf("Розклад:\n");
    printf("День-%d\n Урок1-%s\n Урок2-
%s\n", day1.d, day1.lesson1, day1.lesson2);
    printf("День-%d\n Урок1-%s\n Урок2-
%s\n", day2.d, day2.lesson1, day2.lesson2);
    return 0;
}
```


Існують структури з бітовими полями.

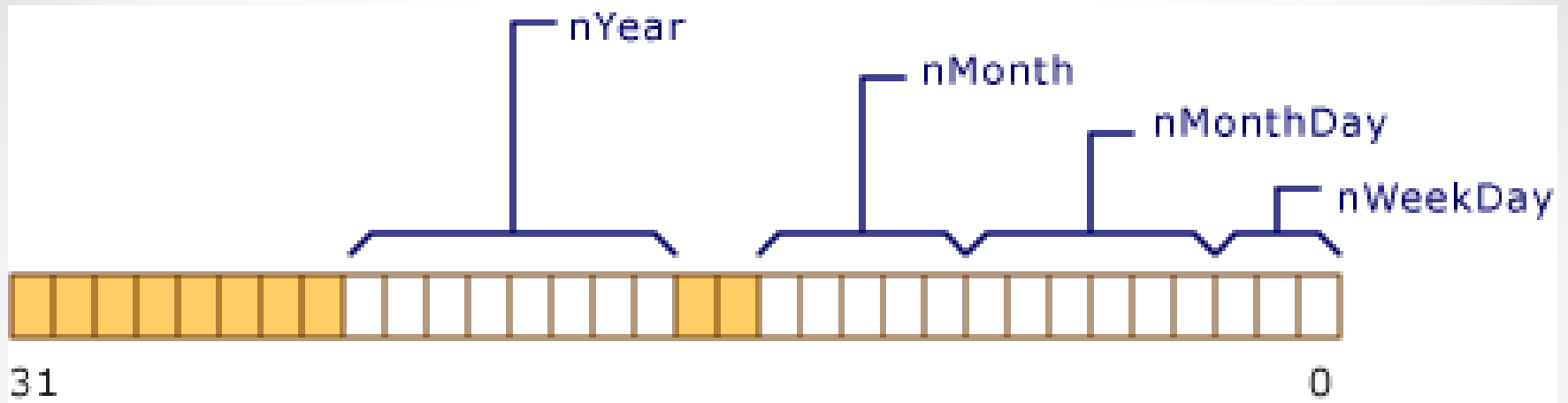
Синтаксис визначення шаблону структури з бітовими полями має вигляд:

```
struct ім'я_шаблону{  
    тип1 назва_змінної1 : r1;  
    тип1 назва_змінної2 : r2;  
    // інші ідентифікатори даних : r3;  
};
```

У наступному прикладі визначається структура date, аналогічна попередньому прикладу, яка містить бітові поля:

```
struct date {  
    unsigned short nWeekDay   : 3;  // 0..7  (3 bits)  
    unsigned short nMonthDay  : 6;  // 0..31 (6 bits)  
    unsigned short nMonth     : 5;  // 0..12 (5 bits)  
    unsigned short nYear      : 8;  // 0..100 (8 bits)  
};
```

На наступному рисунку показана структура пам'яті для **date**.



Поля структур з бітовими полями не можуть бути покажчиками!

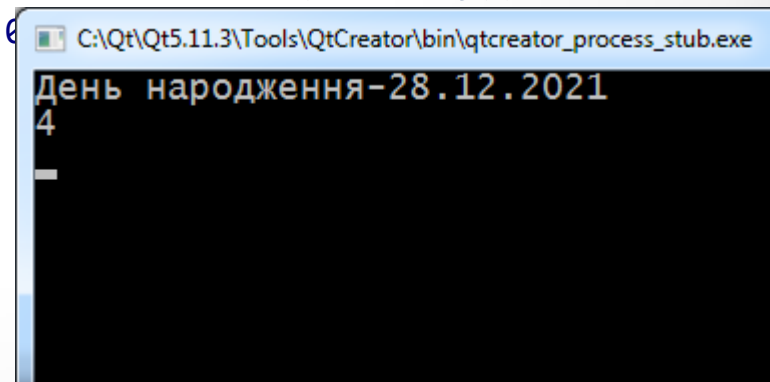
Приклад 5

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

```
enum Days_of_week {
    Mon=1,
    Tue,
    Wed,
    Thu,
    Fri,
    Sat,
    Sun
};
```

```
struct date {
    unsigned short nWeekDay : 3;
    // 1..7 (3 bits)
    unsigned short nMonthDay : 6;
    unsigned short nMonth : 5;
    unsigned short nYear : 8;
};
```

```
int main()
{
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
    //Создание переменной пользовательского
    типа date
    date birthday;
    //Инициализация битовых полей
    birthday.nWeekDay = Mon;
    birthday.nMonth = 12;
    birthday.nMonthDay = 28;
    birthday.nYear = 21;
    printf("День рождения-%d.%d.%d\n",
    birthday.nMonthDay, birthday.nMonth,
    birthday.nYear+2000);
    printf("%d\n", sizeof(birthday));
    return 0;
}
```



Об'єднання

Об'єднання є значення або структура даних, яка може мати кілька різних уявлень.

Об'єднання схожі на структури, з тією різницею, що всі поля розміщуються за однією адресою. Це означає, що розмір об'єднання дорівнює розміру найбільшого його поля.

Об'єднання можуть бути корисні для економії пам'яті при наявності безлічі об'єктів і/або обмеженій кількості пам'яті. Однак для їх правильного використання потрібна підвищена увага.

Загальний синтаксис оголошення шаблону об'єднання має вигляд:

```
union ім'я_шаблону {  
    тип1 назва_змінної1;  
    тип2 назва_змінної2;  
    // інші ідентифікатори даних;  
};
```

Розглянемо приклад. Припустимо, що необхідно до 16-розрядного значенням мати можливість звертатися побайтово. В цьому випадку можна оголосити об'єднання:

```
union data16 {  
    unsigned short word;  
    struct {  
        unsigned char low;  
        unsigned char high;  
    } bytes;  
};
```

Приклад 6

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

```
union MyChar {
    signed char a;
    unsigned char b;
};

union Data16 {
    unsigned short word;
    struct {
        unsigned char low;
        unsigned char high;
    } bytes;
};
```

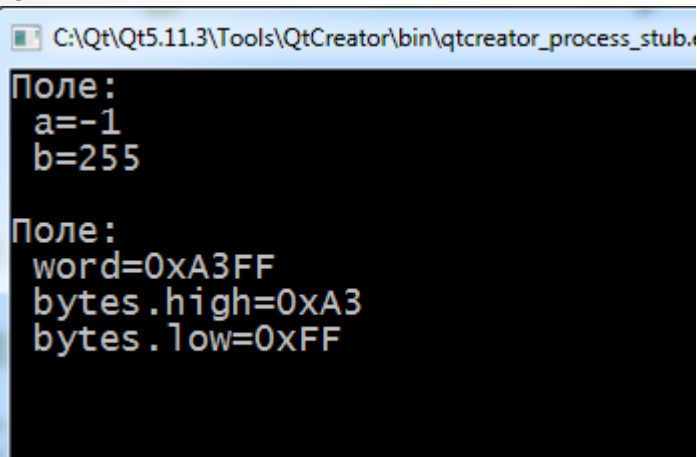
```
int main()
{
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
    //Создание переменной пользовательского
    типа date
    MyChar c;
    Data16 d;

    //Инициализация пользовательских типов
    c.a = -1;
    d.word = 0xA3FF;

    printf("Поле:\n a=%d\n b=%u\n\n", c.a,
    c.b);

    printf("Поле:\n word=0x%X\n
    bytes.high=0x%X\n bytes.low=0x%X\n\n",
    d.word, d.bytes.high, d.bytes.low);

    return 0;
}
```



```
C:\Qt\Qt5.11.3\Tools\QtCreator\bin\qtcreator_process_stub.exe
Поле:
a=-1
b=255

Поле:
word=0xA3FF
bytes.high=0xA3
bytes.low=0xFF
```

Цілі типи даних

Тип	Діапазон значень	Розмір (байт)
signed char	-128 ... 127	1
signed short	-32768 ... 32767	2
signed int		2 або 4
signed long	-2 147 483 648 ... 2 147 483 647	4
signed long long	-9 223 372 036 854 775 807 ... 9 223 372 036 854 775 807	8
unsigned char	0 ... 255	1
unsigned short	0 ... 65535	2
unsigned int		2 або 4
unsigned long	0 ... 4 294 967 295	4
unsigned long long	0 ... 18 446 744 073 709 551 615	8

Цілі типи даних

Модифікатори **signed** та **unsigned** використовуються для зміни способу реалізації зберігання значень.

Формат цілого числа з модифікатором **unsigned**.



Формат цілого числа з модифікатором **signed**.



Цілі типи даних

Розглянемо тип char.

Формат **unsigned char**:



Формат **signed char**:



Цілі типи даних

Діапазон значень змінної залежить від типу та розраховується для цілочисельних типів у відповідності за виразом:

$$D = 0..2^r - 1, \quad \text{для unsigned}$$

$$D = -2^{r-1}..2^{r-1} - 1, \quad \text{для signed}$$

Таким чином, діапазон **unsigned char** – $0..2^8 - 1 \rightarrow 0..255$

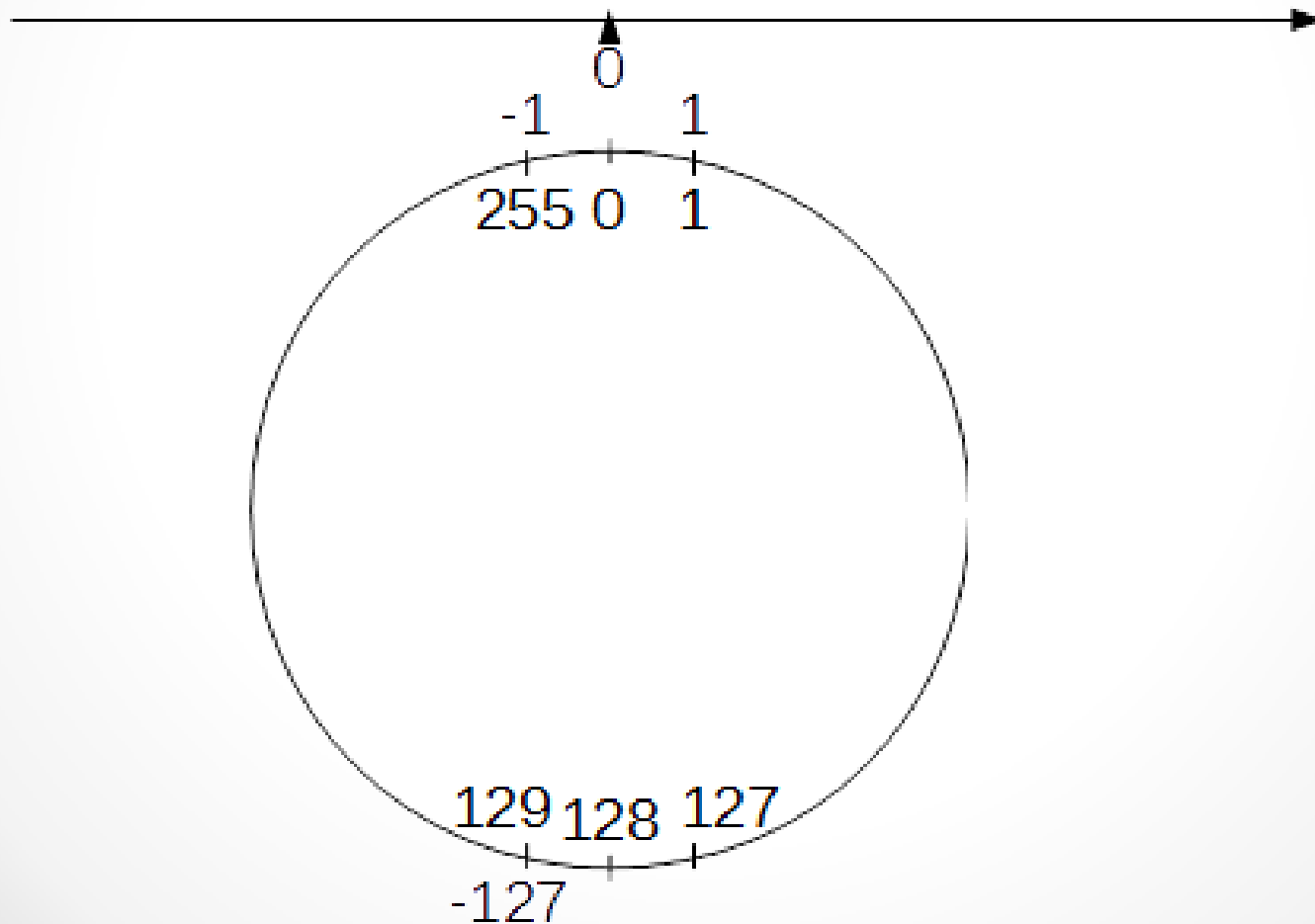
Таким чином, діапазон **signed char** – $-2^7..2^7 - 1 \rightarrow -128..127$

Необхідно відзначити, що числа вісь для цілочисельних типів з лінійної трансформуються у кругову.



Цілі типи даних

Необхідно відзначити, що числа вісь для цілочисельних типів з лінійної трансформуються у кругову. Це відбувається тому, що діапазон чисел обмежується розрядністю осередків пам'яті.



Цілі типи даних

Це призводить до певних помилок при невірному виборі типів змінних.
Наприклад:

```
for(unsigned char i=100; i<=255; i++) {  
    printf("%d\n", i);  
}
```

Приклад 7

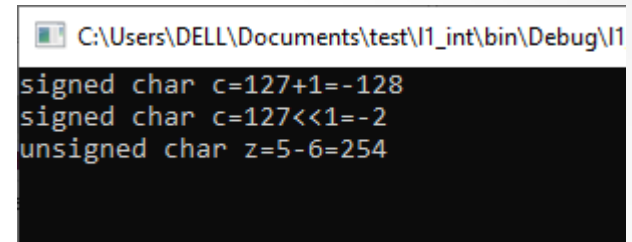
```
#include <stdio.h>
#include <windows.h>

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    unsigned char x, y, z;
    signed char a, b, c;

    a=127;
    b=1;
    c=a+b;
    printf("signed char c=127+1=%d\n", c);

    c=a << 1;
    printf("signed char c=127<<1=%d\n", c);

    x=5;
    y=7;
    z=x-y;
    printf("unsigned char z=5-7=%u\n", z);
    return 0;
}
```

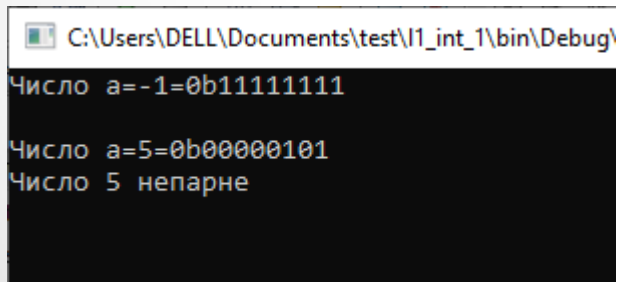


```
C:\Users\DELL\Documents\test\l1_int\bin\Debug\l1
signed char c=127+1=-128
signed char c=127<<1=-2
unsigned char z=5-6=254
```

Приклад 8

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

union char2bit {
    signed char a;
    struct {
        unsigned char b0:1;
        unsigned char b1:1;
        unsigned char b2:1;
        unsigned char b3:1;
        unsigned char b4:1;
        unsigned char b5:1;
        unsigned char b6:1;
        unsigned char b7:1;
    } b;
};
```



```
C:\Users\DELL\Documents\test\I1_int_1\bin\Debug>
Число a=-1=0b11111111
Число a=5=0b00000101
Число 5 непарне
```

```
int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    char2bit c;

    c.a = -1;
    printf("Число a=-1=0b%d%d%d%d%d%d%d%d\n\n", c.b.b7, c.b.b6, c
.b.b5, c.b.b4, c.b.b3, c.b.b2, c.b.b1, c.b.b0);

    c.a = 5;
    printf("Число a=5=0b%d%d%d%d%d%d%d%d\n", c.b.b7, c.b.b6, c.b.b5, c.b.b4, c.b.b3, c.b
.b2, c.b.b1, c.b.b0);
    if(c.b.b0 == 0)
        printf("Число %d парне\n", c.a);
    else
        printf("Число %d непарне\n", c.a);
    return 0;
}
```

Дробові типи даних

Тип	Кількість знаків після коми	Розмір (байт)
float	6-7	4
double	15-16	8
long double	19-20	10

Представлення чисел з плаваючою комою визначається стандартом IEEE 754.

IEEE 754 — широко розповсюджений стандарт формату представлення чисел з плаваючою комою, що використовується як у програмних реалізаціях арифметичних дій, так і в багатьох апаратних (CPU та FPU) реалізаціях. Стандарт визначає формати і методи для арифметики з плаваючою комою в комп'ютерних системах - стандартні та розширені функції для чисел одинарної, подвійної, розширеної і розширюваної точності.

Приклад представлення числа -178,125 в 32-розрядній сітці (тип float).

Зн.	Степень								Мантисса																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Таким чином $-178,125 = 0xC3322000$