

# ТЕМА: Символьні типи даних.

## Тип char

Як розглядали раніше, кожний символ (буква, цифра, знак, службовий) зображений в пам'яті своїм кодом. Тип `char` використовують в основному для зображення символів в системі кодування ASCII, коли коди є однобайтовими числами без знаку. Отже, код може бути в межах 0-255. Для позначення кодів символів використовують тип даних `char` або `unsigned char`. Різниця між цими типами полягає в трактуванні числа, яке означає код символу. Для `char` це є межі від -128 до 127, а для `unsigned char` межі від 0 до 255. Для деяких операцій і функцій це може впливати на результат.

Зауваження. Мова C++ має також тип `wchar_t` для розширених кодів символів розміром 2 байти і відповідні функції їх опрацювання. Для роботи з розширеними символами підключають файли `<wchar>` і `<cwctype>`. [ Ми не розглядаємо ]

Окремі символи позначають одинарними лапками:

```
char c = 'M'; // змінна отримує код ASCII літери 'M'
char d = c + 1; // до коду літери додати 1
cout << c << ' ' << (int)c << endl << d << ' ' << (int)d << endl;
cout << (c > 'K') << endl; // порівняння символів
// M 77
// N 78
// 1
```



Мова C++ дозволяє використовувати літери в арифметичних та інших операціях. Використовують код літери, який трактують як коротке однобайтове ціле число.

Деякі правила кодування в системі ASCII:

- 1) цифри '0'-'9' кодують підряд числами 48-57;
- 2) великі латинські літери 'A'-'Z' кодують підряд числами 65-90;
- 3) малі латинські літери 'a'-'z' кодують підряд числами 97-122;
- 4) літери з кодами 0-31 є службовими, як «перейти до наступного рядка», «табуляція», «клавіша Esc», «звуковий сигнал» тощо;
- 5) кодування кириличних літер починається числом не меншим, ніж 128 – друга половина кодової таблиці; кодування може бути різним залежно від так званої кодової сторінки – наприклад, cp1252.

Переглянути коди літер можна програмно, наприклад, так:

```
// друкувати таблиці кодів літер - окремими групами
for (c='0'; c<='9'; c++) cout << c << ' ' << (int)c << endl; // цифри
for (c='A'; c<='Z'; c++) cout << c << ' ' << (int)c << endl;
    // великі лат.
for (c='a'; c<='z'; c++) cout << c << ' ' << (int)c << endl;
    // малі лат.
int k;
for (k=0; k<=31; k++) cout<<(char)k << ' ' << k << endl; // службові
for (k=128; k<=255; k++) cout << (char)k << ' ' << k << endl;
    // 2-а половина код.табл.
```

Звернути увагу на перетворення типів `int ↔ char`

Всі друковані символи можна записувати в одинарних лапках. Але деякі символи, наприклад, символ переходу до нового рядка, неможливо друкувати на клавіатурі. Для цього передбачені спеціальні керуючі символні константи, наприклад:

\n	новий рядок
\t	горизонтальна табуляція
\"	подвійна лапка
\0	код нуль
\\	обернена коса риска

В цілому запис керуючих констант виглядає як '\n', '\\\'' і т.д.



## Деякі символьні функції (для типу char)

Символьні функції виконують операції з окремим символом. Для роботи з символьними функціями треба, можливо, підключити файл:

```
#include <cctype>
```

Деякі функції:

```
int isalpha(int ch)
```

Якщо аргумент є буквою, функція повертає ненульове значення, інакше повертає нуль:

```
int k1, k2, k3;  
k1 = isalpha('F'); k2 = isalpha(81); k3 = isalpha('+');  
cout << k1 << '\t' << k2 << '\t' << k3 << endl;
```

Буде надруковано:

1	1	0
---	---	---

*Пояснення щодо функції.*

- 1) Ця функція і подібні працюють правильно лише для латинських літер в зв'язку з історичними причинами виникнення.
- 2) Параметрами символьних функцій вважають цілі числа, проте використовують лише їх молодший байт. Параметр-символ – це однобайтове ціле.
- 3) Конкретне повернене ненульове значення може залежати від системи програмування, в даному випадку – число 1.

```
int isdigit(int ch)
```

Якщо аргумент є цифрою, функція повертає ненульове значення, інакше повертає нуль:

```
cout << isdigit('6') << '\t' << isdigit(50) << '\t'
      << isdigit('*') << endl;
//   4           4           0
```

```
int toupper(int ch)
```

Якщо аргумент є буквою, повертає прописний еквівалент, інакше не міняє:

```
cout << (char)toupper('r') << '\t' << (char)toupper('F')
      << '\t' << (char)toupper('8') << endl;
//   R           F           8
```

Приклад задачі. На клавіатурі друкують рядок тексту, який закінчується крапкою  
c1 c2 c3 ... cn .

Кількість символів довільна. Порахувати, скільки цифр є в тексті.

```
char c;  
int k=0; // лічильник цифр  
int all=0; // лічильник всіх прочитаних літер  
do  
{  
    cin.get(c); // наступна літера  
    all++;  
    if (isdigit(c)) k++;  
} while (c != '.');  
cout << k << endl << all << endl;
```

Приклад виконання:

```
data56 : rt78 : s.  
4  
18
```

Зауваження. Читати окремі літери треба не операцією `cin>>c` видобування з потоку, а функцією `get()`, як показано в прикладі. Операція видобування для типу `char` викреслює з потоку пропуски, табуляції та інші подібні літери, а функція `get()` читає всі літери включно з названими.

## Символьні рядки

Символьний рядок – це масив типу `char`, у якому послідовність літер закінчується нульовим байтом:

'р'	'я'	'д'	'о'	'к'	\0				
літери рядка (коди)					нуль	не використано			

Щоб визначити рядок, потрібно оголосити масив символів, передбачивши одну комірку для нульового байта (нульового символу). Всі функції для роботи з рядками розраховані на обмеження рядка нульовим символом, тому він є обов'язковим.

Рядок має змінну довжину.

```
char m1[8] = {'о', 'с', 'і', 'н', 'ь'} ; // це не рядок !!  
char m2[8] = {'л', 'і', 'т', 'о', '\0'} ; // а це рядок !
```

Звернемо увагу, що нульовий символ – це нульовий код, але не цифра '0', яка має код 48. Нульовий символ відіграє фундаментальну роль в рядках, зокрема, для функцій C++.



Безпосередньо записані символьні рядки в подвійних лапках називають символьними рядками-константами:

```
"Print 4*4 matrix"
```

Подібні рядки-константи ми вже використали як коментарі для друкування в попередніх лекціях. До таких рядків-констант компілятор автоматично додає нульовий байт, тому вручну дописувати не потрібно.

Насамкінець зауважимо деяку особливість:

```
char c1 = 'S' ; // правильно
char c2 = "S" ; // неправильно, бо "S" - це рядок з літер
// 'S' і '\0', а точніше - адреса пам'яті, де збережений рядок
```



## Базові операції з рядками

До базових операцій належать:

- ініціалізація масиву символів рядком;
- читання рядка;
- друкування рядка.

Все решту виконують за допомогою функцій роботи з рядками.

Ініціалізацію масиву символів можна робити зразу цілим рядком:

```
char line[7] = "ABCD" ;    // буде ABCD\0\0\0
```

```
char any[] = "Bubbles";    // буде Bubbles\0
```

Не забувати, що розмір масиву повинний врахувати нульовий символ в кінці.

Якщо цього немає, наприклад

```
char tst[5] = "ABCDEFGH" ;    // буде ABCDE
```

і нульовий символ не поміщається, масив не буде рядком, до нього не можна застосувати функцій роботи з рядками.

Читання рядків. Читати рядки з консолі чи файла можна операцією >> видобування, але треба знати, що в цьому разі рядком вважають групу літер до першого пропуску чи табуляції, а після пропуску – це другий рядок:

```
char w1[20], w2[20], w3[50];  
cin >> w1;  // читає до пропуску  
cout << w1 << endl;  
cin >> w2 >> w3;  // наступні рядки - після пропуску  
cout << w2 << endl << w3 << endl;
```

Виконання цієї програми виглядає так:

```
one    two    three  
one  
two  
three
```

Найперший рядок – це вхідні дані, в кінці рядка натиснули Enter.

Треба знати, що виникає аварійна ситуація, якщо вхідний рядок довший, ніж виділений для нього масив, тоді зайва частина рядка перекриє іншу пам'ять програми.

Якщо потрібно коректно читати справді цілий рядок разом з розділовими знаками, використовують функції `getline()` і `get()` потоку `cin`:

`cin.getline(name, num);` // читає рядок разом з Enter  
`name` - ім'я масиву для збереження; `num` - обмежувач кількості символів з урахуванням нульового символу. Потрібний рядок друкують разом з всіма знаками і натискають Enter. Після цього з рядка вибирають перші `num-1` символів і записують в масив `name`, додаючи в кінець нульовий символ. Якщо рядок коротший, ніж `num-1`, то вибирають те, що є. Якщо довший, ніж `num-1`, то решта символів викреслюють:

```
char sent1[15];  
cin.getline(sent1, 15); // читає до 14 символів або до Enter  
cout << sent1 << endl;
```

Результати:

1234 5678

1234 5678

← перше виконання

12345678901234567890

12345678901234

← друге виконання



```
cin.get(name,num) ;
```

Подібно як `getline()`, але решта символів після `num-1` залишаються у вхідному потоці до наступного читання. Символ Enter залишається завжди, тому наступна операція читання може отримати порожній рядок – треба уважно програмувати послідовність операцій.

Приклад:

```
char sent1[15], sent2[15];  
cin.get(sent1,15);  cin.get(sent2,15);  
cout << ':' << sent1 << endl << ':' << sent2 << endl;
```

Отримаємо:

```
12345678901234567890  
:12345678901234  
:567890
```

або:

```
ab cd ef  
:ab cd ef  
:
```

Друкування рядків. Рядки можна друкувати так, як у прикладах: `cout<<масив`. Можна друкувати також посимвольно, використовуючи вказівники:

```
char text[30] = "What's your name?\n";  
char *p = text;  // вказівник на масив  
while ( *p )  cout << *p++;
```

## Типові функції для роботи з рядками

Для програмування задач опрацювання рядків переважно використовують стандартні бібліотечні функції. Повний перелік функцій є в документації. Найбільш поширені є такі:

<code>strcpy(s1, s2)</code>	копіює рядок <code>s2</code> в рядок <code>s1</code> : $s1 \leftarrow s2$
<code>strncpy(s1, s2, n)</code>	копіює перші $n$ символів рядка <code>s2</code> в рядок <code>s1</code> : $s1 \leftarrow s2[0:n-1]$
<code>strcat(s1, s2)</code>	дописує копію рядка <code>s2</code> в кінець рядка <code>s1</code> (конкатенація): $s1 = s1 + s2$
<code>strlen(s1)</code>	обчислює довжину рядка <code>s1</code> в символах
<code>strcmp(s1, s2)</code>	порівнює лексикографічно рядки <code>s1</code> і <code>s2</code> ; повертає 0, якщо однакові, від'ємне число, якщо $s1 < s2$ , і додатне, якщо $s1 > s2$
<code>strchr(s1, ch)</code>	повертає вказівник на позицію першого наявного символу <code>ch</code> в рядку <code>s1</code> ; якщо <code>ch</code> немає – повертає нульовий вказівник
<code>strstr(s1, s2)</code>	повертає вказівник на позицію першого підрядка <code>s2</code> в рядку <code>s1</code> ; якщо <code>s2</code> немає – повертає нульовий вказівник

Зауважимо, що програміст має сам гарантувати правильний вибір розміру `s1` для перших трьох функцій, а також подібних до них.

Для розв'язування задач щодо рядків треба скласти алгоритм з використанням стандартних функцій. Стандартні функції виконують більшу частину роботи.

Для використання бібліотеки функцій роботи з рядками, можливо, буде потрібно підключити файл `#include <cstring>`.

*Приклад задачі. Речення складається з слів, розділених одним пропуском, і закінчується крапкою, наприклад "one\_two\_three\_four.". Надрукувати в стовпчик окремі слова. Побудувати нове речення в оберненому порядку слів.*

Алгоритм. Замість крапки в кінці записати пропуск, тоді всі слова будуть закінчені пропуском. Використовуючи символ пропуску, шукати кожне слово і копіювати в масив слів. Надрукувати масив. Сполучити слова в речення, переглядаючи масив в оберненому порядку.

Схема пошуку окремих слів заданого речення може бути такою:

o	n	e	_	t	w	o	_	t	h	r	e	e	_	f	o	u	r	_	\0
↑			↑	↑			↑												
pb	-	-	pe	pb	-	-	pe												
перше слово (початок і кінець слова)				друге слово (пересунути вказівники)				третє слово						і т.д.				до \0	



```
// приклад задачі на рядки
#include <iostream>
using namespace std;
// речення поділити на слова і побудувати в оберненому порядку
int main()
{
    char fsent[200] = "one two three four." ; // задане речення
    // речення можна читати: cin.getline(fsent,200);
    char bsent[200] = ""; // тут буде обернений порядок слів,
                        // спочатку - порожній рядок !
    // wordlist - масив слів - статична матриця
    // (на практиці - масив вказівників на слова):
    char wordlist[30][20]; // не більше 30 слів,
                        // не довших 19 букв кожне - без контролю
    int k=0; // лічильник знайдених слів
    char *pb, *pe; // пара вказівників межі чергового слова
    // крапку в кінці заданого речення замінити пропуском
    fsent[strlen(fsent)-1] = ' ';
                        // індекси літер від нуля, тому мінус 1
    // cout << fsent << endl << strlen(fsent) << endl;
    // можна для контролю
```

```
// ділимо задане речення на слова
pb = fsent; // перша буква першого слова
while ( *pb ) // поки не досягли нульового символу
{
    pe = strchr(pb, ' ');
    // пошук пропуску, починаючи від pb кожного слова !!
    strncpy(wordlist[k], pb, (pe-pb)); // копіюємо знайдене слово
    wordlist[k][pe-pb] = '\0'; // вручну дописуємо нульовий символ
    k++; // збільшити лічильник знайдених слів
    pb = pe + 1; // перша літера наступного слова або кінець речення
}

// друкуємо список знайдених слів
for (int i=0; i<k; i++) cout << wordlist[i] << endl;

// будуємо речення в оберненому порядку слів
for (int i=k-1; i>=0; i--)
{ strcat(bsent,wordlist[i]); strcat(bsent," "); }
cout << bsent << endl;
system("pause");
return 0;
}
```

Результати:

```
one  
two  
three  
four  
four three two one
```





## Задачі і алгоритми

*Задача 1. Маріо, герой комп'ютерної гри, прямуючи на рівні 1-1 до кінця, має зійти на "напівпіраміду" з блоків, перед тим як доскочити (якщо він хоче отримати якнайбільше очок) до прапорця. Нижче наведено знімок екрану цього моменту.*



*Напишіть програму, яка відтворює цю напівпіраміду, використовуючи решітки (#) замість блоків. Висота піраміди є параметром – число, не більше 20. Наприклад:*

```
height: 8
  ##
 ##
###
####
#####
#####
#####
#####
#####
```

*Зауважте, що дві праві колонки блоків мають бути однакової висоти.*

Задача 2. Шифр Цезаря полягає в тому, щоб циклічно зсунути алфавіт на деякий ключ, який визначає кількість літер, на які робиться зсув (можна переглянути [http://en.wikipedia.org/wiki/Caesar\\_cipher](http://en.wikipedia.org/wiki/Caesar_cipher) ). Якщо  $p$  - деякий звичайний текст (тобто незашифрований),  $p_i$  -  $i$ -а літера в  $p$ ,  $k$  - ключ (невід'ємне ціле число), то кожну літеру  $c_i$  в шифрованому тексті  $c$  обчислюють так:

$$c_i = (p_i + k) \% 26$$

Якщо зсув сягає кінця алфавіту, то переходимо по колу від  $z$  до  $a$ , чи від  $Z$  до  $A$ .

Задано:

- текст  $p$ , до якого входять малі і великі латинські літери, пропуски, розділові знаки;
- розмір тексту  $n$ ;
- ключ  $k$ , який може бути як меншим 26, так і більшим 26.

Скласти програму для шифрування тексту.

Символи, які не є буквами алфавіту, треба залишити без змін.

Великі букви мають залишитись великими, а малі – малими.

Надрукувати текст  $p$  і текст  $c$ .

Приклад. Для  $k=13$  і тексту

Be sure to drink your Ovaltine!

маємо отримати

Or fher gb qevax lbhe Binygvar!

Задача 3. Скласти програму, обернену за змістом до задачі 2, тобто виконати дешифрування заданого зашифрованого тексту.

Задано:

- зашифрований текст  $c$ ;
- розмір тексту  $n$ ;
- ключ  $k$ , який може бути як меншим 26, так і більшим 26.

Скласти програму для дешифрування тексту за такими самими правилами, як в задачі 2. Визначити, якою буде в цьому випадку формула обчислення  $p_i$ .

