Тема заняття: Розробка і виконання програм з використанням операторів циклів для опрацювання табличних і рядкових величин.

План.

1. Рядкові величини
2. Символьні рядки.
3. Оголошення символьних рядків у програмі.
4. Ініціалізація символьного рядка.
5. Передавання рядків у функції.
6. Вказівники і символьні рядки.
7. Функції для роботи з рядками.
8. Символьні рядки.
9. Приклад програми

Комп'ютер використовують для обробки інформації різних видів, зокрема, і текстової. Текст можна розглядати як сукупність символьних рядків. Розглянемо засоби, які мова С++ надає для роботи з ними.

Оголошення символьних рядків у програмі.

Символьні рядки у С++ являють собою звичайні масиви. Головна відмінність між символьними рядками й іншими типами масивів полягає в тому, що символьний рядок можна виводити, наприклад, командою cout << abc;, а не організовувати цикл для поелементного виведення даних.

Кінець символьного рядка позначається символом NULL, що у програмі зображується як спеціальний символ '\0'. Можна сказати, що саме NULL перетворює частину символьного масиву в символьний рядок.

Для оголошення символьного рядка усередині програми просто оголосіть масив типу char з кількістю елементів, достатньою для зберігання необхідних символів. Наприклад, оголосимо змінну з ім'ям A, здатну зберігати 32 символи (не забувайте, що символ NULL є одним із цих 32 символів):

char A[32];

Це оголошення створює масив з елементами, пронумерованими від A [0] до A [31]. A[0] A[1] A[2] ... A[30] A[31].

С++ трактує символьний рядок як масив типу char. Розглянуті раніше програми використовують рядкові константи, уміщені у подвійні лапки:

"Це рядкова константа"

При створенні рядкової константи компілятор С++ автоматично додає символ NULL \0.

Коли при виконанні програми за допомогою вихідного потоку cout виводиться рядкова константа, cout використовує символ ' \ 0' як ознаку кінця рядка.

Більшість функцій C++, призначених для роботи з рядками, використовують символ NULL для визначення останнього символу рядка.

У наступній програмі уводиться рядок з використанням циклу for, а потім виводиться на екран за допомогою cout:

```
#include<iostream.h>

#include<conio.h>

int main()

{

char abc  [27]; // 26 символів плюс NULL

char let;

int i;

//Заповнення рядкового масиву латинськими літерами

for  (let =  'A',  i = 0;  let <=  'Z';  let++, i++)

abc  [i]  = let; abc  [i]  =  ' \0'; //Ознака кінця рядка

cout << abc; //Виведення рядкової константи

getch(); return 0;

}
```

У програмах на C++ зустрічаються окремі символи, взяті в одинарні лапки (наприклад, 'A'), і символи у подвійних лапках ("A"). Між цими записами є принципова різниця. Символ в одинарних лапках являє собою символьну константу. Компілятор C++ виділяє тільки один байт пам'яті для її зберігання. Однак символ у подвійних лапках являє собою рядкову константу - зазначений символ і символ NULL (доданий компілятором). Таким чином, компілятор виділить два байти для символьного рядка.

Ініціалізація символьного рядка.

Як ви вже знаєте, C++ дозволяє ініціалізувати масиви при оголошенні. Символьні рядки C++ не є винятком. Для ініціалізації символьного рядка при оголошенні вкажіть необхідну послідовність символів усередині подвійних лапок: char tit [32] = "Вивчаємо мову C++";

Якщо кількість символів, що присвоюється рядку, менша від розміру масиву, більшість компіляторів C++ присвоюють символи NULL елементам рядкового масиву, що залишаються. Як і у випадку з масивами інших типів, якщо ви не вказуєте розмір масиву, що ініціалізуєте при оголошенні, компілятор C++ виділить достатньо пам'яті для розміщення зазначених символів рядка і службового символу NULL:

```
char tit[] = "Вивчаємо мову C++";
```

При компіляції наведеного фрагменту для рядка tit буде виділено 18 байтів (15 зображених символів + 2 пропуски + символ '\0').

У наступній програмі символьні рядки ініціалізуються при оголошенні:

```
#include<iostream.h>

#include<conio.h>

int main()

{

char tit [32]  = "Hello, C++";

char les[] = "Simvolni ryadki!";

cout<<"Book:   "<<tit<<endl;

cout<<"Lesson:   "<<les<<endl;

getch() ; return 0;

}
```

Зверніть увагу, що для масиву tit виділено 32 байти, хоча рядок, занесений в нього при ініціалізації, - коротший. Тому пізніше, в програмі, у цей масив можна буде вмістити довший рядок, обов'язково помістивши в кінці символ '\0'.

Передавання рядків у функції.

Передавання символьного рядка у функцію подібне до передавання будь-якого масиву як параметру. Усередині функції потрібно просто вказати тип масиву (char) і квадратні дужки масиву. Не треба вказувати розмір рядка. Наприклад, наведена програма використовує функцію show_ryad для виведення символьного рядка:

```
#include<iostream.h>

#include<conio.h>

void show_ryad(char ryad[])
```

```
{
cout << ryad << endl;
}
int main()
{
show_ryad("Hello, C++!");
show_ryad("Good Bye, C++!");
getch(); return 0;
}
```

Як бачите, функція show_ryad трактує параметр символьного рядка як масив:

```
void   show_ryad(char ryad[])
```

Оскільки символ NULL вказує кінець рядка, функція не вимагає параметра, що задає кількість елементів у масиві. Замість цього функція може визначити останній елемент, просто знайшовши в масиві символ NULL.

Як ви вже знаєте, функції C++ часто використовують символ NULL для визначення кінця рядка. Наступна програма містить функцію з ім'ям str_len, що шукає у рядку символ NULL для визначення кількості символів у ньому. Далі функція використовує оператор return для повернення значення довжини рядка до функції, що викликала. При виконанні програми кілька різних символьних рядків передаються у функцію, і на екран виводиться довжина кожного з них:

```
#include<iostream.h>
#include<conio.h>
int str_len(char ryad[])
{
int i;
for (i = 0; ryad[i]   !=  '\0'; i++);
return(i);
}
int main()
{
```

```
char title[]  ="Vchimosya programuvannyu";

char lesson[]="Simvolni ryadki";

cout<<"Ryadok:   "<<title<<" mae "<< str_len(title)<<"  simvoliv"<<endl;

cout<<"Ryadok:"<<lesson<<"mae"<<str_len(lesson) <<"simvoliv"<<endl;

getch(); return 0;

}
```

Як бачите, у функції організовано перегляд рядка, починаючи з першого символу (елемент з індексом 0), і при виявленні NULL перевірка припиняється. На цей момент змінна і дорівнює кількості символів у рядку, тому саме її значення повертають оператором return (i);.

Далі ви побачите, що наведена функція моделює одну зі стандартних функцій для роботи з рядками.

Вказівники і символьні рядки.

Другий спосіб визначення рядка - це використання вказівника на символ. Оголошення char *b; задає змінну b, що може містити адресу деякого об'єкта. Однак у цьому випадку компілятор не резервує місце для зберігання символів і не ініціалізує змінну b конкретним значенням. Зробити це можна, наприклад, присвоївши b вказівник на вже існуючий символьний масив або динамічно виділивши пам'ять під новий масив:

```
#include<iostream.h>

#include<conio.h>

int main()

{

char Ryad[]="Hello,  world!"; //оголошуємо символьний масив

char *b;

b = &Ryad[7]; //оголошуємо вказівник на символ тепер b вказує на 8-й символ

Ryad *b = ‘W’; //присвоюємо першому елементу b символ W

cout << b; //виводимо рядок b на екран (World!)

getch(); return 0;

}
```

Функції для роботи з рядками.

Заголовковий файл <string.h> містить набір функцій, які забезпечують ефективну роботу з рядками. Розглянемо деякі з них.

char* strcat(char*Ryad1, const char*Ryad2);

- об'єднує рядки Ryad1 і Ryad2 і записує результат у рядок Ryad1.

char* strcpy(char*Ryad1, const char*Ryad2);

- копіює рядок Ryad2 у рядок Ryad1.

int strlen(const char* Ryad);

- повертає довжину рядка (кількість символів). Символ '\0' не враховується.

int strcmp(const char*Ryad1,const char*Ryad2);

- порівнює рядки Ryad1 і Ryad2. Повертає 0, якщо рядки рівні, число менше від нуля, якщо Ryad1 < Ryad2 і число більше від нуля, якщо Ryad1 > Ryad2.

char* strlwr(char*Ryad);

- перетворює великі літери рядка на малі (обробляє тільки латинські букви).

char* strupr(char*Ryad);

-перетворює малі літери рядка на великі (обробляє тільки латинські букви).

char* street (char*Ryad, char Simvol);

- заповнює рядок зазначеним при виклику функції символом.

 char* strrev(char* Ryad);

- міняє порядок літер у рядку на протилежний.

Приклад:

Дослідження функцій для роботи з рядками.

```
#include<string.h>
#include<iostream.h>
#include<conio.h>
#include<windows.h>
char*Ukr(const char* text);//Прототип функції
int main()
{
char Ryad1[50];
```

```
char Ryad2[50];
cout<<Ukr("Уведіть 1-е слово:");
cin>>Ryad1;
cout<<Ukr ("Уведіть 2-е слово:");
cin>>Ryad2 ;
cout<<Ryad1<<"    " <<Ryad2<<endl ;
//Обчислення довжини рядків
int x=strlen(Ryad1); int y=strlen(Ryad2);
cout<<" Ryad1="<<x<<" Ryad2 = "<<y<<endl;
//Об'єднання рядків strcat(Ryad1, Ryad2);
cout<< "Ryad1=" <<Ryad1<<" Ryad2 = "<<Ryad2<<endl ;
//Перетворення малих літер рядка на великі
strupr(Ryad1); strupr(Ryad2);
cout<<"Ryad1="<<Ryad1<<" Ryad2="<<Ryad2<<endl ;
//Перетворення великих літер рядка на малі
strlwr(Ryad1);s
trlwr(Ryad2);
cout<<"Ryad1="<<Ryad1<<" Ryad2="<<Ryad2<<endl ;
//Копіювання рядка 2 у рядок 1
strcpy(Ryad1, Ryad2);
cout<<"Ryad1="<<Ryad1<<" Ryad2 = "<< Ryad2<<endl;
//Заповнення рядка 2 зірочками
street  (Ryad2#   1 *.') ?
cou t<<"Ryad2 ="<<Ryad2 << end1;
//Зміна порядку символів у рядку на протилежний
cout<<strrev (Ryad1) <<endl ; getch();return 0;
}
// функція для використання українських літер char bufUkr  [256];
char*Ukr(const char* text)
```

{

CharToOem(text, bufUkr); return bufUkr;

}

Дослідивши цю програму, ви навчитеся працювати з рядковими величинами в С++. А підключивши бібліотеку <windows.h> ви зможете використати український алфавіт, «пропустивши» рядок через функцію CharToOem ().

Всі символи для обробки у комп'ютері подаються у вигляді числових кодів, які разом складають кодову таблицю. Коли комп'ютер порівнює два рядки, він насправді порівнює коди відповідних символів у рядках, починаючи з перших. За кодами перших двох неоднакових символів робиться висновок про результат порівняння. Наприклад, «ABCD» < «ABCE», «ABC» < «ABCD».

Перед написанням програми розглянемо ще одну функцію С++. Досить часто виникає потреба уводити в масив цілий рядок тексту. З цією метою використовують функцію cin.getline. Зверніть увагу, що назва функції складається з двох частин, відокремлених крапкою. Це означає, що функція getline є методом об'єкту cin.

Функція cin.getline вимагає три аргументи: масив символів, у якому буде зберігатися рядок тексту, довжина й символ-обмежувач. Наприклад, у такому фрагменті програми:

char ryad1[32];

cin.getline (ryad1, 32, '\n');

оголошується масив ryad1 з 32 символів, а потім із клавіатури в цей масив зчитується рядок тексту. Функція припиняє зчитування символів у випадках:

- якщо зустрічається символ-обмежувач '\n';
- якщо уводиться вказівник кінця файлу;
- якщо кількість уведених символів виявляється на один менше, ніж зазначене в другому аргументі (останній символ у масиві резервується для завершального нульового символу).

Якщо зустрічається символ-обмежувач, він зчитується й відкидається. Третій аргумент cin.getline має '\n', як значення за замовчуванням, так що попередній виклик функції міг бути написаний у такому вигляді: cin.getline (ryad1, 32);

#include<iostream.h>

#include<string.h>

```
#include<conio.h>
int main()
{
int len; // довжина рядка
char r[81]; // місце зберігання рядка
char *r1,*r2;
cout<<"Vvedi 1 ryadok:   ";
cin.getline(r, 80); // уведення першого рядка
len = strlen(r); // визначення довжини рядка
r1 = new char[len +1];  // динамічне виділення
//пам'яті під рядок r1 strcpy(r1,  r); //копіювання уведеного
//рядка в рядок r1
cout<<"Vvedi 2 ryadok:   " ;
cin.getline(r,  80); //уведення другого рядка
len * strlen(r);
r2 = new char[len +1];       //динамічне виділення
//пам'яті під рядок r2
strcpy(r2,  r) ;
if(strcmp(r1,  r2)  > 0) //який з уведених рядків більший?
cout<<"Ryad r1: \t"<<r1<<" \n\t>\n"<< "Ryad r2 : \t"<<r2«endl; else if(strcmp(r1,
r2)  == 0)
cout<<"Ryad rl: \t" <<r1<<" \n\t=\n"<< "Ryad r2 : \t"<<r2«endl;
else
cout<<"Ryad r1: \t"<<r1<<" \n\t<\n"<<"Ryad r2:\t"<<r2<<endl;
// видалення рядків з пам'яті delete  []rl; delete []r2;
getch();return 0;
}
```