**Практична робота №7. Розгляд прикладів програм з символьними та рядковими величинами, масивами та структурами**

**Мета:** навчитися складати програми з символьними та рядковими величинами, масивами та структурами**.**

**Завдання до практичної роботи надається в кінці теоретичної частини.**

**Строк виконання 29.03.2021**

**ТЕОРЕТИЧНА ЧАСТИНА**

Для обробки символьних типів даних бібліотека функцій **string.h** має велику кількість вбудованих функцій, що збільшують продуктивність праці програмістів та скорочують час на розробку програм, наприклад:

* функції перевірки рядків;
* функції маніпулювання рядками.

Функції наводяться у вигляді списків, що згруповані за їх розташуванням у заголовних файлах. Найчастіше надаються прототипи функцій, що описують, як слід використовувати функції у програмах.

Розглянемо ще деякі важливі функції використання рядків типу string.

На **Лекції 9** ми розглянули (перед виконанням завдання перегляньте приклади в лекції).

***Функції визначення довжини рядка:***

* **str.size();**
* **str.length();**
* **str.max\_size(); .**

***Функції додавання одного рядка або його частини до іншого рядка:***

* **str.append(st);** — додає рядок **st** до кінця рядка **srt;**
* **str.append(st,k,n);** — додає до рядка str n символів рядка **st,** починаючи з позиції **k.**

***Функція включення рядка в рядок:***

* **str.insert(k,st)** — вставляє в рядок **str** з позиції **k** рядок **st**;
* **str.insert(k1,st,k2,n)** — вставляє в рядок **str** з позиції **kl** n символів рядка **st**,починаючи з позиції **k2** в рядку **st**.

Наприклад:  
*// вставка подстроки в строку*  
**string str, st = “string”;**  
**str = “My text”;**  
**str.insert(3,st);**  
**cout << “str = “<< str << endl;**

Результат: **str = My string text**  
**string str, st = “string “;**  
**str = “My text”;**  
**str.insert(3,st,2,4);**  
**cout << ” str = ” << str << endl;**

Результат: **str = My ring text**

***Функція вилучення символів із рядка:***

* **str.remove(k,n)** — вилучає **n** символів з рядка **str**, почи­наючи з позиції **k**.

Наприклад:  
**string str = “My text string”;**// удаление подстроки из строки  
**str.remove(3,4);**  
**cout << “str = “<< str << endl;**

Результат виконання:  
**str = My string.**  
***Функція заміни частини рядка або усього рядка:***

* **str.replace(st)** — заміняє рядок **str на st**;
* **str.replace(k,n,st)** — заміняє в рядку **str n** символів, починаючи з позиці**ї k** рядка **st**;
* **str.replace(kl,nl,st,k2,n2)** — заміняє в рядку **str nl** сим­волів з позиції **kl** частиною в**n2** символи рядка **st**, починаючи з позиції **k2**.

Наприклад:  
// *замена подстроки в строке*  
**string str, st = “substring”;**  
**str =”My text string”;**   
**str.replace(8,9,st);**  
**cout <<“str = “<< str<<endl;**

Результат виконання:  
**str = My text substring**  
 **string str, st = ” substring”;**  
**str=”My text long”;**  
**str.replace(3,5,st,l,3);**  
**cout <<” str = “<< str<<endl;**

Результат виконання:  
**str = My sublong**  
  
***Функція обміну змістом двох рядків:***

* **str.swap(st)** — обмінює зміст рядків **str та st**.

Наприклад:  
*//————- обмен содержимым строк*  
**string str, st=”Coлнечная погода”;**   
**str = “Дождливая погода”;**  
**str.swap(st);**  
**cout <<” str = “<<str<<” st = “<<st<<endl;**

Результат виконання:  
**str = Солнечная погода st = Дождливая погода.**

***Функція виділення частини рядка:***

* **str.substr(k,n)** — повертає частину рядка **str** в **n** симво­лів, починаючи з позиції **k.**

Наприклад:  
**string str, st;** *//————- выделение части строки* **str=”Moй текст string “;  
st=str. substr(4,13);  
cout << ” st = ” << st << endl;**

Результат виконання:  
**st = текст string.**

***Функція пошуку позиції входження підрядка в рядок:***

* **str.find(st,k)** — шукає зліва граничну позицію входжен­ня рядка **st** в рядок **str**,починаючи з **k**-ої позиції рядка **str;**
* **str.rfind(st,k)** — шукає справа граничну позицію входження рядка **st** в рядок **str**,починаючи з **k**-ої позиції рядка **str.**

Наприклад:  
// *позиция вхождения подстроки в строку*  
**#include <iostream>**

**#include <string>**

**#include <conio.h>**

**using namespace std;**

**void main ()**

**{ string str, st="as";**

**str="as large as life!" ;**

**int p=str.find(st,0);**

**cout <<" p="<<p<<endl;**

**p=str.rfind(st,str.length());**

**cout <<" p="<<p<<endl;**

**getch ();**

**}**Результат виконання:  
**р=0 та р=9.**

***Функція присвоювання усього рядка або його частини іншому рядку:***

* **str.assign(st)** — присвоює весь рядок **st** типу **string** або масив **char[ ]** — рядку **str**типу **string;**
* **str.assign(st,k,n)** — присвоює **n** символів рядка **st** рядку **str,** починаючи з **k**-ої позиції.

Ці функції можна використовувати для перетворення рядка типу **char** у рядок **string**. Наприклад:  
*//————————– присваивание строк*  
**char st[ ]=”Пришло жаркое лето!”;**  
**string str;**  
**cout<<” str = “<< str.assign(st,7,12)<<endl;**

Результат виконання:  
**str = жаркое лето!**або  
**string str, st=”O, лето красное, любил бы я тебя!”;  
cout<<“str = “<< str.assign(st,3,12) << endl;**

Результат виконання:  
**str = лето красное.**

***Функція порівняння рядків або їхніх частин:***

* **str.compare(st)** — порівнює рядки **st** та **str** і повертає значення:

**<0 — якщо st < str;**  
**=0 — якщо st = str;**  
**>0 — якщо st > str;;**

* **str.compare(st,k,n)** — порівнює n символів рядка **st** з рядком **str,** починаючи з **k**-ої позиції. Наприклад:

Оскільки тип string є класом, то, щоб порівняти два рядки між собою можна використати операцію **‘= =’**. Якщо два рядки однакові, то результат порівняння буде true. У противному випадку, результат порівняння буде false.

Але, якщо потрібно порівняти частину одного рядка з іншим, то для цього передбачена функція compare().

Функція працює наступним чином. Якщо викликаючий рядок менше рядка s, то функція повертає -1 (від’ємне значення). Якщо викликаючий рядок більше рядка s, функція повертає 1 (додатне значення). Якщо два рядка рівні, функція повертає 0.

**Приклад**. Демонстрація роботи функції compare():

// тип string, функція compare()

string s1 = "012345";

string s2 = "0123456789";

int res;

res = s1.compare(s2); // res = -1

res = s1.compare("33333"); // res = -1

res = s1.compare("012345"); // res = 0

res = s1.compare("345"); // res = -1

res = s1.compare(0, 5, s2); // res = -1

res = s2.compare(0, 5, s1); // res = -1

res = s1.compare(0, 5, "012345"); // res = -1

res = s2.compare(s1); // res = 1

res = s2.compare("456"); // res = -1

res = s2.compare("000000"); // res = 1

***Функція перетворення рядка типу string у рядок типу char:***

* **str.c\_str()** — перетворює рядок типу **string** у рядок типу **char.**

**Розглянемо приклади використання наведених функцій при обробці рядків типу string.**

***Приклад 1.*** Увести список прізвищ і відсортувати його за алфавітом

**#include <iostream>**

**#include <string>**

**using namespace std;**

**int main()**

**{**

**const int n=5;**

**string spis[n];**

**int i, k;**

// Введення прізвищ

**for (i = 0; i < n; i++)**

**{ cout <<"\*\*\* Enter "<<(i+1)<<" name\n";**

**cin >> spis[i]; };**

// Сортування прізвищ

**for (k = 1; k < n; k++)**

**for (i = 0; i < n-k; i++)**

**if (spis[i] > spis[i+1]) spis[i].swap(spis[i+1]);**

**//** Виведення результатів

**cout<<"\n\*\*\*\*\* Rezult spisok\n";**

**for(i = 0; i < n; i++)**

**cout << (i+1) << ' ' << spis[i] << endl;**

**return 0;**

**}**

Результати виконання:  
\*\*\*\*\* Enter 1 name   
Котин С. Б.  
\*\*\*\*\* Enter 2 name   
Бобров М. Т.  
\*\*\*\*\* Enter 3 name   
Авдеев P. Л.  
\*\*\*\*\* Enter 4 name   
Попов Г. Д.  
\*\*\*\*\* Enter 5 name   
Елкин П. P.  
\*\*\*\*\* Rezult spisok  
1   Авдеев P. Л.  
2   Бобров M. Т.  
3   Елкин П. Р.  
4   Котин С. Б.  
5   Попов Г. Д.

Для порівняння прізвищ у програмі використовується звичайна операція «>», а для взаємозаміни — функція **swap()** (

***Приклад 2.*** Вести до пам’яті комп’ютера будь-який текст, відокремити в ньому всі слова, вивести їх на екран та визначити найдовше слово

#include <iostream>

#include <string>

using namespace std;

//---------------------------------------------------------------------------

int main()

{

string txt, sl, slmax; // slnax – найдовше слово

int i, k=0, n=0, max=0;

//---------------- Введення тексту

cout <<"\*\*\*\*\* Enter text\n";

getline(cin,txt,'\n');

for (i = 0; i <= txt.length()-1; i++)

{ if ((txt[i]==' ') || (txt[i]==',') || (txt[i]=='.'))

{n++; // n —лічильник пропусків та знаків пунктуації

if (n>1) continue; k++; //k — лічильник слів

cout<< k <<" word - "<< sl<<" = "<<sl.size() << " symbols\n";

//--------------- визначення слова з найбільшою кількістю літер

if (sl.size() > max)

{ max = sl.size();

slmax=sl; }

sl=""; //очищення sl для нового слова

}

else {n=0; sl=sl+txt[i]; } //змінна s1 — формує слово

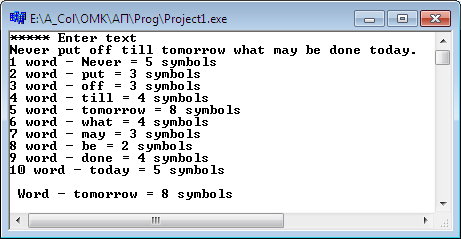
}

cout<<"\n Word - "<< slmax << " = " <<slmax.size()<<" symbols\n";

return 0;

}

Результат:



**Приклад 3.** З уведеного списку прізвищ (без ініціалів) вибрати такі, що починаються на задані літери і мають задане закінчення, та вивести повідомлення про прізвище з найменшою кількістю літер.

#include <iostream>

#include <string>

#include <conio.h>

#include <Windows.h>

using namespace std;

//---------------------------------------------------------------------------

int main()

{

system("color F0"); // Встановлення кольору фону для С+Builder

const int n = 6;

string spis [n], pok, p ; // Список прізвищ, задані закінчення та перші літери

int i, minfam, k=0; // додаткові змінні для збереження проміжних результатів

int lpok, lspis, lp; //

//------------------------------ Введення списку прізвищ

cout << "\*\*\*\*\* Enter "<<n<<" name";

for (i = 0; i < n; i++)

{ cout << "\n" << i+1 << '\t';

cin >> spis [i]; }

//------------------------------ Введення перших літер

cout << "\*\*\*\*\* Enter first letter\n";

cin >> p;

lp = p.size(); // визначення довжини перших літер

//------------------------------ Введення закінчення

cout << "\*\*\*\* Enter last letters\n";

cin >> pok;

lpok = pok .size();// визначення довжини закінчення

// Пошук прізвищ із заданими літерами

cout << "\*\*\*\*\* Search name\n";

for (i = 0; i < n; i++)

{ lspis = spis [i].size(); // довжина прізвища

// spis[i].substr(0,lp) - це вибираються початкові літери прізвища

// spis[i].substr(lspis-lp,lp) це вибираються кінцеві літери прізвища

if (spis[i].substr(0,lp) ==p && spis[i].substr(lspis-lpok,lpok) == pok )

cout<<spis [i]<<endl; }

//-------------- Пошук прізвища з найменшою кількістю літер

minfam = spis[0].size(); // початкове значення з першого прізвища

for (i = 1; i < n; i++)

if (spis[i].size() < minfam)

{ minfam = spis[i].size();

k=i; }

//-------------- Виведення результатів пошуку

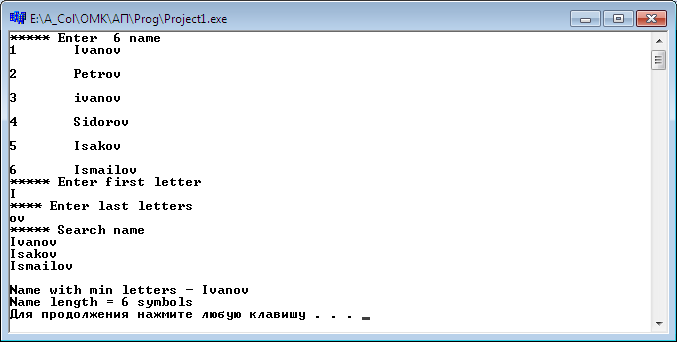
cout<<"\nName with min letters - "<<spis [k]<<" \n";

cout << "Name length = " << spis [k].size() << " symbols \n";

system("pause");

return 0;

}

Результати обчислень:  


У програмі для зберігання перших літер використовується змінна з ім’ям **р**, а для закінчення — змінна **роk**. Визначення перших літер та закінчення прізвища здійснює функція **size**. У цілому в операторі **if**… визначається як перші літери, так і закінчення прізвища. Змінна **minfam** існує для знаходження прізвища з найменшою кількістю літер згідно з алгоритмом визначення мінімального елемента масиву.

**Контрольні запитання для письмових відповідей та завдання**

1. Які оператори введення-виведення символьних масивів ви знаєте? В яких випадках вони використовуються?
2. Як визначається кінець рядка?
3. Надайте список функцій перевірки рядків, які розглянуті на двох останніх заняттях.
4. Надайте список функцій маніпулювання рядками, які розглянуті на двох останніх заняттях.
5. Розгляньте приклад 3. Чи є у Вас зауваження щодо пошуку прізвища з найменшою кількістю літер? Чи всі прізвища будуть виведені, якщо ні, то що, на Вашу думку потрібно зробити?
6. В C++Shell Скласти програму оброблення рядків типу string**.**
7. Ввести для двох рядків такі речення,

Better late than never.

It is never too late to learn.

1. Визначте довжину кожного рядка і виведіть на екран разом з відповідним текстом.
2. Напишіть, які функції потрібно було б застосувати, щоб
3. об’єднати два речення,
4. знайти у першому реченні позицію слова «than»,
5. поставити перед словом «than» слово «learn».

**Визначення та використання структур (Лекція 9)**

**Структура** — це сукупність різнотипних елементів, яким присвоюється одне ім’я (воно може бути відсутнім), що займає одну ділянку пам’яті. Елементи, що складають структуру, називаються полями.

Змінна типу структура, як і будь-яка змінна, повинна бути описана. Цей опис складається з двох кроків: опису шаблону (тобто складу) або типу структури та опису змінних структурного типу.

Синтаксис опису структури має вигляд:

**struct [<ім’я структури>]**  
**{ <тип 1> ім’я поля 1;**  
**<тип 2> ім’я поля 2 . . .;**  
**} р1, р2 . . .;**

де **struct** — службове слово;

**<ім’я структури>** — ім’я типу структура (може бути відсутнім);

**<тип 1>, <тип 2>** — імена стандартних або визначених типів;

**ім’я поля 1, ім’я поля 2,…** — імена полів структури;

**р1, р2 . . .;** — імена змінних типу структура.

Наприклад, для знаходження середнього бала, отриманого студентами в період сесії з дисциплін «Математика», «Фізика» та «Програмування», визначимо таку структуру:

**struct stud**

**{ char fam [25];**            // фамилия и инициалы

**int mat, fiz, prg;**          // предметы

**float sb;**                   // средний балл

**}** **st1, st2;**

Змінні **st1 і st2** можна оголосити окремим оператором, наприклад:

**struc stud stl st2;.**

Ініціювання полів структури слід здійснювати або при її описі, або в тілі програми. При описі структури ініціювання полів виглядає, наприклад, так:

**struct stud**

**{ char fam [25];**

**int mat,fiz, prg;**

**float sb;}**

**st1 = {"Кравченко И. С.", 4, 5, 5};**

**st2 = {"Тесленко А. М.", 3, 4, 5};**

Якщо ініціювання виконується в тілі програми, то для звернення до імені поля треба спочатку записати ім’я структурної змінної, а потім ім’я поля. Ці обидва записи відокремлюються крапкою і являють собою складене ім’я.

Отже, у випадку появи змінної **st1** у програмі для її ініціювання можна записати

**stud st1** **= {“Кравченко И. С.”, 4, 5, 5};** або ініціювання виконується за допомогою складених полів. Розглянемо ілюстраційну програму:

**#include <iostream.h>**

**#include <string.h>**

**#include <stdio.h> // для puts: int puts(const char \*str)**

**#include <conio.h>**

**using namespace std;**

**main ( )**

**{ struct stud**     //----- описание структуры

**{ char fam [20];**

**int mat, fiz, prg;**

**float sb;**

**} st1, st2;**

**strcpy (st1.fam, "Кравченко И. С.");**

**st1 .mat = 4;**

**st1 .fiz = 5;**

**st1 .prg = 5;**

**st1.sb = float (st1.fiz + st1.mat + st1.prg)/3;**

**st2 = st1;**

**puts (st2.fam);**      //---------------- вывод фамилии

**cout << st2.mat << st2.fiz << st2.prg << st2.sb << endl;**

**getch();**     //задержка экрана вывода результата

**}**

У наведеній програмі організується присвоювання всім полям структури **st1** відповідних значень.

**char strcpy (s, \*st);** — виконує операцію копіювання байтів рядка **st** у рядок **s** (включаючи  **“\0”**; повертає **s**)

Структурна змінна **st2** того ж типу, що і **st1**, тому справедлива операція **st2 = st1;.**

Якщо функція використовує тільки один структурний тип, то цей тип можна оголосити без імені. Тоді раніше розглянуту структуру можна оголосити таким чином:

**struct**

**{ char fam [25];**

**int mat, fiz, prg;**

**float sb;**

**} stl, st2;**

Коли при описі структур у деякій функції або в межах видимості змінних у різних функціях є багато (але не всі) однакових полів, то їх слід об’єднати в окрему структуру. Її можна застосовувати при описі інших структур, тобто поля структури можуть самі бути типу **struct**. Це називається **вкладеністю структур** — її можна використати, наприклад, якщо треба обробляти списки студентів та викладачів університету. Студентські списки містять дані: прізвище та ініціали, дата (день, місяць, рік) народження, група та середній бал успішності, а в списках викладачів присутні такі дані: прізвище, ініціали, дата народження, кафедра, посада. У процесі обробки списку студентів і списку викладачів можна оголосити відповідно такі структури:

**struct stud**

**{ char fio [25];**

**int den, god;**

**char mes [10];**

**char grup;**

**flout sb; }**

**struct prep  
{ char fio [25];  
int den, god;  
char mes [10];  
char kaf, dolg;  
}**

В оголошених типах однакові поля можна об’єднати в окрему структуру і застосовувати її при описі інших типів. Поетапно це виглядає так:

* загальна структура —

**struct spd**

**{ char fio [25];**

**int den, god;**

**char mas[10]; }**

* структура для опису інформації про студентів —

**struct stud**

**{ spd dr;**

**char grup;**

**float sb}**

**st1, st2;**

* структура для опису інформації про викладачів —

**struct prep**

**{ spd dr;**

**char kaf [10];**

**char dolg [15];**

**} pr1, pr2;**

У структурах **stud і prep** для оголошення поля, що містить дані про прізвище і дату народження, використовується раніше описаний тип **spd**. Тепер до поля **fio, den, god, mes**можна звернутися, використовуючи запис **st1.dr.fio**, наприклад, при зверненні до функції введення:

**gets (st1.dr.fio);** або    **gets (pr1.dr.fio);**

**Пояснення.**

Прототип функції gets: (б-ка cstdio) char \* gets( char \* string );

Отримати рядок зі стандартного потоку введення. Функція gets зчитує символи зі стандартного потоку введення до символу нового рядка n або доки, поки не буде досягнутий кінець файлу EOF, після чого зберігає лічені символи в рядок типу char.

Символ нового рядка n не копіюватиметься в рядок.

Нульовий символ автоматично додається після останнього копійованого символу в string, щоб сигналізувати про кінець рядка.

Функція gets дещо відрізняється від функції fgets, по-перше повертає рядок зі стандартного потоку введення в якості аргументу, а, по-друге, кінцевий символ нового рядка не входить до її складу . на відміну від fgets. А ще, gets не дозволяє встановити обмеження на кількість зчитувальних символів, тому потрібно бути обережними з розміром масиву, на який вказує параметр str, щоб уникнути переповнення буфера.

Параметри: **string**Покажчик на масив типу char, в якому зберігається рядок.

Значення, що повертається. У разі успіху, функція повертає той же параметр, string. Якщо відбувається помилка, повертається нульовий покажчик.

Використовуйте функції ferror або feof для перевірки внутрішніх станів показників помилки або кінця файлу – EOF.