**Практична робота №7.Написання програм з використанням символьних та рядкових величин**

**На період** роботи в дистанційній формі навчання потрібно використовувати **онлайн компілятор C++ Shell, який доступний за адресом** [**http://cpp.sh**](http://cpp.sh) (інструкція надана після тексту ЛР№3). При наявності на домашньому комп’ютері іншого завдання можна виконати на ньому. Результати надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді URL-посилання для робіт, виконаних у **C++ Shell,** або для робіт, на компіляторі С++ домашнього комп’ютера cpp-файл з іменем у форматі

**АР<Номер групи><Номер практичної / лабораторної [літера позначення типу роботи P – практична, R – лабораторна]<Прізвище англійською>**. Наприклад, **АРPTBD-2107P**buts.срр.

При відсутності можливості використовувати компілятор або відсутності можливості доступу до Інтернету текст програми набрати в Блокноті або WordPad Windows, в крайньому випадку текст програми можна записати у зошити, сфотографувати, оформити відповідним чином (включити титульний лист, записати завдання, а потім включити файли з фотографіями) та надіслати на електронну адресу викладача.

Оформлені роботи повинні надсилатися у вигляді текстового документу з іменем файлу:

**АР<Номер групи><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище англійською>**. Наприклад, **АРPTBD-2104P**buts.doc.

**Тему в заголовку листа записати**

**АР <Номер групи>-><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище >**

**Строк відсилки ЛР 19.04.2024.**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, тему в заголовку листа записати

**АР<Номер групи>-Запитання-<Прізвище>**.

**Мета**: навчитися писати програми з використанням символьних та рядкових величин, структур та масивів **на мові С++** в **консольному режимі.**

**Методичні вказівки щодо організації самостійної роботи студентів**

1. Прочитати (повторити) основні теоретичні відомості. Переглянути лекцію №9 "Символьні та рядкові величини. Складні типи даних – масиви, структури..
2. Запустити середовище програмування С++ .
3. Записати програму, що виконує 3 завдання з пп.4.1-4.3 (за завдання 4.1 мінімальна оцінка 2 бали, за 4.2 та 4.3 будуть надані додаткові бали, по 3 за кожне). В першому рядку кожної програми записати

*// ПТБД-21 Група № Прізвище*

вказавши номер своєї групи та своє прізвище.

Як і в ПР №5, програма повинна запитати номер завдання (число 1, 2, 3 або 0 для закінчення) і в залежності від введеного значення виконувати відповідне завдання. Якщо введений 0 – програма припиняє роботу.

Вхідні дані ввести, а результати вивести, використовуючи потокове введення-виведення даних.

1. Завдання:
   1. Написати програму, яка питає ім`я, порівнює з тими, що вона має як елементи символьного масиву (рядки типу string) та вітає або повідомляє, що “не знайома”.

Елементи символьного масиву задати через ініціалізацію (див. приклад в кінці теоретичних відомостей).

* 1. В заданому рядку "Don’t cut the bough you are sitting on." знайти і вивести найкоротше і найдовше слово, вказати номери позицій, з яких вони починаються. ***Вказівки:*** перегляньте потрібні функції (пошук символу, виділення підрядка, знаходження довжини рядкової змінної). Вам може буди потрібно ввести проміжні рядкові зміні, а для організації перегляду в циклі булеву змінну. Продумайте алгоритм.
  2. Розробити програму, яка запитує текстовий рядок, і в текстовому рядку замінює будь-яку кількість однакових символів, що йдуть один за одним підряд на один такий же символ та цифру, яка відповідає кількості видалених символів (Наприклад: ‘*1CABk3KKK111DeFf0100fk0cccccc*’=*’1CABk3K212DeFf0101fk0с5*’). ***Вказівки:*** початковий рядок вводити з клавіатури; отриманий рядок виводити під початковим рядком.

**Запитання**

1. Що таке масив та які існують різновиди масивів?
2. Як здійснюється звернення до елементів масивів?
3. Що таке рядки та значення елементів символьного типу?
4. Що являє собою масив символьного типу?
5. Як здійснюється введення символьних даних?
6. Як можна в програмі визначити рядок символів?
7. Як виконується порівняння даних символьного типу?
8. Як визначити кількість символів у рядку?
9. Як описуються дані типу структура?
10. Які типи полів може містити структура?
11. Назвіть функції визначення довжини рядка.
12. Назвіть функції додавання одного рядка або його частини до іншого рядка.
13. Яка функція використовується для включення рядка в рядок.
14. Яка функція використовується для вилучення символів із рядка.
15. Яким чином можна замінити частини рядка або увесь рядок?
16. Як обміняти зміст двох рядків?
17. Яка функція використовується для виділення частини рядка?
18. Як здійснити пошук позиції входження підрядка в рядок?
19. Як порівняти рядки або їхні частини?
20. Яка функція виконує перетворення рядка типу string у рядок типу char?

**Основні теоретичні відомості**

Рядок – це послідовність символів, яка закінчується нульовим байтом *‘\0’*. Дана послідовність символів записується в подвійних лапках.

Послідовності в С++ представляються масивами або покажчиками, тому стандартно програми мовою C++ зберігають рядки як масиви типу *char*. Для оголошення строкової змінної необхідно всередині програми оголосити масив типу *char* з кількістю елементів, достатньою для зберігання необхідного рядка символів.

Головна відмінність між символьними рядками та іншими типами масивів полягає в тому, що C++ визначає останній елемент масиву. Програми мовою C++ представляють кінець символьного рядка за допомогою символу NULL, який в C++ зображується як спеціальний символ '\0', або нульовий байт.

Створення рядків, наприклад, відбувається наступним чином:

char str2[10]="Hello";

Для стандартної роботи з символьними рядками в стилі С++ призначена бібліотека string.h, яка містить наступні основні функції:

– **char \**strcat*(char \*dest, const char \*src);** – дописує рядок *src* в кінець *dest*;

– **char \**strncat*(char \*dest, const char \*src, size\_t n);** – дописує не більше *n* початкових символів рядка *src* (або весь *src*, якщо його довжина менше) у кінець *dest*;

– **char \**strchr*(const char \*str, int ch);** – шукає символ *ch* у рядку *str*, починаючи з голови, і повертає його адресу, або NULL якщо символ не знайдено;

– **char \**strrchr*(const char \*str, int ch);** – шукає символ *ch* у рядку *str*, починаючи з хвоста і повертає його адресу, або NULL якщо символ не знайдено;

– **int *strcmp*(const char \*str1, const char \*str2);** – лексикографічне порівняння рядків;

– **int *strncmp*(const char \*str1, const char \*str2, size\_t n);** – лексикографічне порівняння перших *n* байтів рядків;

**– char \**strcpy*(char \*dest, const char \*src);** – копіює рядок з *src* у *dest*;

– **char \**strncpy*(char \*dest, const char \*src, size\_t n);** – копіює до *n* байт рядку з *src* у *dest*;

– **size\_t *strlen*(const char \*str);** – повертає довжину рядка;

– **size\_t *strspn*(const char \*str, const char \*alp);** – визначає максимальну довжину початкового підрядка з *str*, що складається виключно з байтів, перерахованих в *alp*;

– **char \**strpbrk*(const char \*str, const char \*alp);** – знаходить перше входження будь-якого символу, перерахованого в *alp*, у *str*;

– **char \**strstr*(const char \*str, const char \*pstr);** – знаходить перше входження рядка *pstr* у *str*.

Однак окрім розглянутого вище стандартного способу роботи з символьними рядками в бібліотеці STL С++ є ще одна бібліотека для роботи з рядками. Вона дозволяє легко вирішувати наступні задачі з високою ефективністю:

– створювати, присвоювати, копіювати і видаляти рядки;

– виконувати перетворення типів символьних змінних;

– порівнювати рядки;

– поєднувати рядки;

– визначати довжину рядка;

– знаходити і заміщати потрібний фрагмент у рядку.

Для використання даної бібліотеки необхідно підключити заголовний файл <string>. Після цього операція створення нового рядка виявиться настільки ж простою, як і створення змінної будь-якого базового типу.

**string** hi(“hello”);

**string** lo=“greetings”;

**string** es=””;

Рядковим змінним можна присвоювати значення, як і змінним будь-яких інших типів:

**string** name(“Fred”);

name = “Paul”;

При цьому всі операції розподілу пам'яті будуть виконані коректно.

Операції + (конкатенація), операція порівняння (==, > тощо) дозволяють легко виконувати відповідні дії над рядками.

**Приклад програми з використанням масиву рядкових величин.**

**#include <iostream>**

**#include <string>**

**#include <Windows.h>**

**using namespace std;**

**//-------------------------------------------------------**

**int main()**

**{**

**system("color F0");**

**string month[5]={"april","may","june","july","august"};**

**string name;**

**bool found=false;**

**cout << "What month? ";**

**cin >> name;**

**for (int i=0;i<5;i++)**

**if (name==month[i])**

**{cout << "Hello, " << name << "!\n";**

**found=true;}**

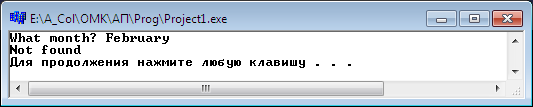
**if (!found ) cout << "Not found " <<endl;**

**system("pause");**

**return 0;**

**}**

**Результат роботи програми**



**Деякі зауваження щодо роботи з символьним типом даних char**

***Тип даних char***

Змінна типу char займає 1 байт. Однак, замість конвертації значення типу char в ціле число, воно інтерпретується як ASCII-символ.

**ASCII** (від англ. “**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange”) — це американський стандартний код для обміну інформацією, який визначає спосіб представлення символів англійської мови (+ декілька інших) у вигляді чисел від 0 до 127. Наприклад: код букви 'а' — 97, код букви 'b' — 98. Символи завжди поміщаються в одинарні лапки.

Символи від 0 до 31 в основному використовуються для форматування виводу. Символи від 32 до 127 використовуються для виведення. Це літери, цифри, знаки пунктуації, які більшість комп’ютерів використовує для відображення тексту (англійською мовою).

Наступні два оператори виконують однакові дії - присвоюють змінним типу char ціле число 97:

**char ch1(97);** // ініціалізація змінної типу char цілим числом 97

**char ch2('a');** // ініціалізація змінної типу char символом 'a' (97)

При використанні фактичних чисел для представлення символів (з таблиці ASCII) потрібна уважність. Наступні два оператори виконують різні дії:

**char ch(5);** // ініціалізація змінної типу char цілим числом 5

**char ch('5');** // ініціалізація змінної типу char символом '5' (53)

***Оператор static\_cast***

Якщо ви хочете вивести символи у вигляді цифр, а не у вигляді букв, то потрібно повідомити cout виводити змінні типу char в вигляді цілочисельних значень. Можна присвоїти змінній типу int змінну типу char і вивести її, але це не дуже хороший спосіб:

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**char ch(97);**

**int i(ch); // присвоюємо значенням змінної ch змінній типу int**

**cout << i << endl; // виводимо значення змінної типу int**

**return 0;**

**}**

Результат:

97

Кращим способом є конвертація змінної з одного типу даних в інший за допомогою **оператора static\_cast**.

**Синтаксис static\_cast виглядає наступним чином:**

static\_cast<новий\_тип\_даних>(вираз)

static\_cast приймає значення з (вираз) в якості вхідних даних і конвертує його у вказаний вами <новий\_тип\_даних>.

Приклад використання оператора static\_cast для конвертації типу char в тип int:

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**char ch(97);**

**cout << ch << endl;**

**cout << static\_cast<int>(ch) << endl;**

**cout << ch << endl;**

**return 0;**

**}**

Результат виконання програми:

a  
97  
a

**static\_cast** приймає (вираз) в якості вхідних даних. Якщо ми використовуємо змінну (в виразі), то ця змінна змінює свій тип тільки в інструкції з оператором static\_cast. Процес конвертації ніяк не впливає на вихідну змінну з її значенням! В наведеному вище прикладі, змінна ch залишається змінною типу char з колишнім значенням, чому є підтвердженням останній оператор з cout.

В static\_cast немає ніякої перевірки діапазону, тому при використанні занадто великих або занадто маленьких чисел для конвертованого типу, то відбудеться **переповнення**.

***Введення символів***

Наступна програма просить користувача ввести символ. Потім вона виводить цей символ і його ASCII-код:

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**cout << "Input a keyboard character: ";**

**char ch;**

**cin >> ch;**

**cout << ch << " has ASCII code " << static\_cast<int>(ch) << endl;**

**return 0;**

**}**

Результат виконання програми:

Input a keyboard character: q  
q has ASCII code 113

Зверніть увагу, що навіть якщо cin дозволить вам ввести декілька символів, змінна ch буде зберігати тільки перший символ (саме він і розміщується у змінній). Інша частина користувацького введення залишиться у вхідному буфері, який використовує cin і буде доступна для використання наступним викликам cin.

Приклад:

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**cout << "Input a keyboard character: ";**

**// припустимо, що користувач ввів "abcd"**

**char ch;**

**cin >> ch;**

**// ch = 'a', "bcd" залишається у вхідному буфері**

**cout << ch << " has ASCII code " << static\_cast<int>(ch) << endl;**

**// Звернуть увагу, що наступний cin не просить**

**// користувача що-небудь ввести, дані підтягуються з**

**// вхідного буферу!**

**cin >> ch; // ch = 'b', "cd" залишається в буфері**

**cout << ch << " has ASCII code " << static\_cast<int>(ch) << endl;**

**return 0;**

**}**

Результат виконання програми:

Input a keyboard character: abcd  
a has ASCII code 97  
b has ASCII code 98

***Розмір, діапазон і знак типу сhar***

В С++ для змінних типу char завжди виділяється 1 байт. За замовчуванням, char може бути як **signed, так і unsigned** (хоча зазвичай signed). Якщо ви використовуєте char для зберігання ASCII-символів, то вам не потрібно вказувати знак змінної (так як і signed, і unsigned можуть містити значення від 0 до 127).

Але якщо ви використовуєте тип char для зберігання невеликих цілих чисел, то тоді слід уточнити знак. Змінна типу char signed може зберігати числа від -128 до 127. Змінна типу char unsigned має діапазон від 0 до 255.

***Що використовувати: ‘\n’ чи endl (std::endl)?***

При використанні cout, дані для виведення можуть поміщатися в буфер, тобто cout може не відправляти дані відразу ж на виведення. Замість цього він може залишити їх при собі на деякий час. Це робиться в цілях підвищення продуктивності.

І '\n', і endl обидва переносять курсор на наступний рядок. Тільки endl ще гарантує, що всі дані з буферу будуть виведені, перед тим, як продовжити.

Коли використовувати '\n', а коли endl?

Використовуйте endl, коли потрібно, щоб ваші дані виводилися відразу ж (наприклад, під час запису в файл або при оновленні індикатора стану будь-якого процесу). Зверніть увагу, це може спричинити за собою незначне зниження продуктивності, особливо якщо запис на пристрій відбувається повільно (наприклад, запис файлу на диск).

Використовуйте '\n' у всіх інших випадках.

***Різниця між одинарними і подвійними лапками при використанні з символами***

Символи завжди поміщаються в одинарні лапки (наприклад, 'а', '+' чи '5'). Змінна типу char представляє тільки один символ (наприклад, літеру а, символ + чи число 5). Некоректним записом буде:

**char ch('56'); // змінна типу char може містити тільки 1 символ**

Текст, який знаходиться в подвійних лапках, називається рядком (наприклад, "Hello, world!"). **Рядок** (**тип string**) — це набір послідовних символів.

Ви можете використовувати літерали типу string в коді:

**cout << "Hello, world!";**

**// "Hello, world!" - це літерал типу string**

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.

***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

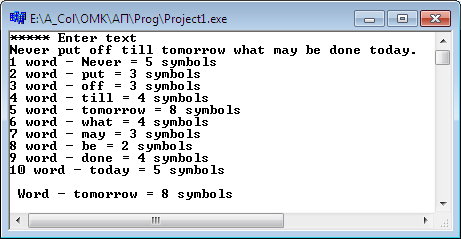
}

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

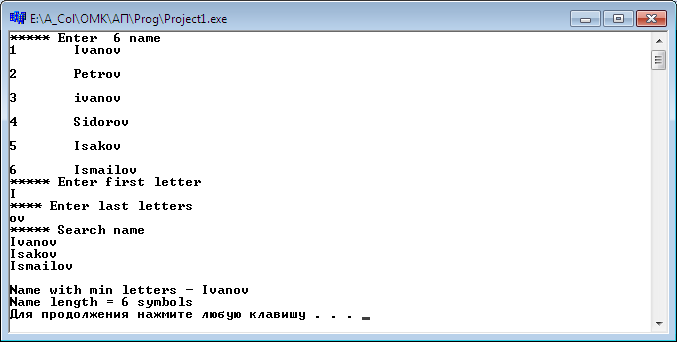
cout<<"\nName with min letters - "<<spis [k]<<" \n";

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**

**(№№1-4 по 0,5 балів, решта – 1 бал)**

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».