**Лекція № 11.** **Покажчики, символьні та рядкові величини.**

**ДАНІ СИМВОЛЬНОГО ТИПУ**

Для роботи з даними символьного типу використовуються масиви, які більш детально будемо розглядати в наступних лекціях. Попередньо надамо визначення.

Опис масивiв у програмі відрізняється від опису простої змінної наявністю після імені квадратних дужок **«[ ]»**, в яких задається кількість елементів масиву (розмірність).

**У мові C++ нумерація елементів масиву починається з 0.**

**<тип> <ім’я> [n];**

**<тип> <ім’я> [n] = {значення};**

**<тип> <ім’я> [  ] = {значення}; /\*** масив відразу, можна не вказувати його розмір \*/

Часто застосовуються багатовимірні масиви. У них позиція елемента визначається записом декількох індексів. Найбільш розповсюджені **двовимірні масиви** або матриці.

Двовимірний масив є списком одновимірних масивів. Наприклад, для оголошення двовимірного  масиву цілочисельних значень розміром 10×20 з іменем num, необхідно записати:

int num[10][20];

У мові програмування C++ кожна розмірність записується у власній парі квадратних дужок.

Щоб отримати доступ до елемента  масива num 3×5, потрібно використовувати запис num[3][5].

На рис. 1 зображено двовимірний масив 8х8 елементів.

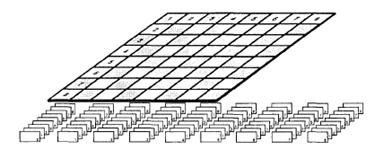


Рисунок 1 Представлення двовимірного  масиву

Масив задається або списком елементів у тому порядку, и якому вони розташовані у пам’яті, або подається як масив масивів, кожний з яких поміщається в свої фігурні дужки**«{}»**. При оголошенні і одночасному ініціюванні багатовимірних масивів можна опускати кількість індексів тільки першого виміру. Якщо ініціювання не здійснюється під час оголошення масиву, то кількість індексів треба вказувати явно.

**Рядок** (або ще кажуть "рядкова змінна або рядкова величина") являє собою масив символів, який закінчується нуль-символом. Нагадаємо, що нуль-символ має код, що дорівнює **0**, і запис у вигляді керуючої послідовності **‘\0’**. За розташуванням нуль-символу визначається фактична довжина рядка. Кількість елементів символьного масиву складається з кількості символів у рядку плюс **1**, тому що нуль-символ також є елементом масиву.

Адреса першого символу рядка може використовуватися по-різному:

* якщо рядок застосовується при ініціюванні масиву типу **char**, адреса його першого елемента стає синонімом імені масиву. Наприклад, ідентичними є такі описи масиву:
* **char st [  ] = "Слово";**
* **char st [6] = “Слово”;**
* **char st [6] = {‘С’ ‘л’ ‘о’ ‘в’ ‘о’ ‘\0’};**

При описі символьного масиву його ім’я — не змінна, а покажчик-константа на початок рядка, тому її не можна використовувати в деяких операціях адресної арифметики. Зокрема, не можна здійснювати операцію присвоювання вигляду:

**char st [20];**

**st = “Петренко”;** — **запис неправильний**, тому що не можна змінити значення **st**.

**Покажчики** — це змінні, котрі містять адресу пам’яті, розподіленої для об’єкта відповідного типу. При оголошенні змінної-покажчика слід вказати тип даних, адресу яких буде містити змінна, та ім’я покажчика з символом «\*».

Загальний формат опису покажчика має вигляд:

**тип \*ім’я;**

де **тип** — тип значень, на який вказує покажчик;  
**ім’я** — ім’я змінної-покажчика;  
«\*» — операція над типом, що читається «покажчик на тип».

Наприклад:

**int \*рn** – покажчик на ціле значення;  
**float \*pf1, \*pf2;** — два покажчики на дійсні значення.

Покажчики не прив’язують дані до якого-небудь визначеного імені змінної і можуть містити адреси будь-якого неіменованого значення. Існує адресна константа **NULL**, що означає порожню адресу.

**Перед використанням покажчика у програмі його обов’язково необхідно ініціювати**, іншими словами, необхідно присвоїти адресу якого-небудь даного, інакше можуть бути непередбачені результати.

Для одержання доступу до значення змінної, адреса якої зберігається в покажчику, досить у відповідному операторі програми записати ім’я покажчика з символом «\*» — здійснити операцію розіменування.

**Виконання дій з елементами символьного масиву** здійснюється через індекси або через покажчики. Покажчики ми розглядати в цьому курсі не будемо. Для доступу до будь - якого символу рядка використовується індекс масиву **char**. Тобто, якщо описана змінна **char str [3];**, то третім елементом ма­сиву можна скористатися, записавши: **str [2]**.

Для роботи з символьними масивами використовують масиви покажчиків, кожен з елементів яких містить адресу рядка масиву даних у пам’яті. Такий спосіб дозволяє зберігати дані з рваними краями, наприклад, деяку текстову інформацію. Масив з «рваними» краями схожий на двовимірну таблицю, рядки якої можуть мати різну довжину. Використання масиву покажчиків (**char \*fio[ ]**) для збереження рядків дозволяє заощаджувати пам’ять, а процес обробки рядків виконується значно швидше, бо змінюються тільки покажчики, а не вміст рядків.



Рисунок 1. Приклад масиву з "рваними" краями

**Приклад 1**. Виведення даних з масиву з "рваними" краями

**#include <iostream>**

**using namespace std;**

**main ( )**

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

**char \*fio[ ] = { "Petrenko",**

**"Golovko",**

**"Korz",**

**"Kutz",**

**"Ushko",**

**"Plush" }; *//*** *ініціалізація масиву покажчиків*

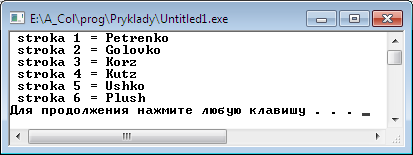
**int str;**

**for (str = 0; str <=5; str++)**

**cout << " stroka " << (str + 1) << " = " << \*(fio + str) << endl;**

**system("pause"); // для затримки екрану виведення результатів**

**}**

Результати виконання програми:  


**У процесі роботи з елементами двовимірного масиву** застосовують також індекси масиву. Якщо описаний список прізвищ **char°spis°[5]°[15];**, то для використання символу масиву слід записати: **spis [і][j]**.

Аналогічно, якщо оголошений масив покажчиків **char \*str [5]**, що містить 5 елементів, кожний з яких вказує на рядок, то доступ до символу рядка можна здійснити з використанням запису **\*(str [і] + j)**.

**Введення рядків** можна здійснювати різними способами, найбільш розповсюдженими з яких є:

* введення шляхом ініціювання при оголошенні символьних масивів:

**char st [15] = “Диск”;**  
**char st [  ] = “Диск”;**

У цьому випадку двовимірні масиви можна ініціювати по-різному, наприклад, у вигляді:

**char str [5][20] = {“Петренко И. И. “, “Головко С. С. “, . . . ,};**  
**char str [ ][20] = {“Петренко И. И. “, “Головко С. С. “, . . . ,};**

* використання потокового введення **сіn >>**. Здійснюється у випадку, коли рядок не містить пропусків, тому що символ пропуску є роздільником введення даних, наприклад:

**char st [5];  сіn >> st;**  
**char str [5][20];  сіn >> str [i];**

* посимвольне введення за допомогою функції **get( )**, наприклад: **get (st[i]);**
* введення за допомогою функції **cin.get**:

**cin.get (str[i], size, endl);,**

де **size** — кількість символів, що зчитуються;

* введення з використанням функції **cin.getline**:

**cin.getline (str[i], sizeof (str[i]-l));,**  
де **sizeof()** — функція визначення розміру рядка.

Виведення рядкових даних реалізується з використанням стандартного вихідного потоку **cout**:

**cout << st;**

**cout.write(st, size);** тощо.

Для потокового введення-виведення доцільно застосовувати функції **cout.setw(w),setprecision(d), cout.width(w) і cout.precision(d).**

Введення-виведення символьних масивів можна здійснити за допомогою відповідних функцій заголовного файла **stdio.h.**, наприклад:

* для введення рядків — **gets(st); та scanf (% s,st);**
* для виведення рядків **— puts(st); і printf(% s,st);.**

ранніх версіях С++ рядки розглядалися як символьні масиви. Для роботи з ними розроблено бібліотеку функцій **string.h,** що містить ефективні засоби для роботи з рядками. Згодом була розроблена стандартна бібліотека шаблонів **Standard TemplateLibrary (STL),** яка надає більш потужні засоби, об’єднані в клас **string**.

Для обробки символьних типів даних бібліотека функцій **string.h** має велику кількість вбудованих функцій:

* функції перевірки символів;
* функції перетворення символів;
* функції перевірки рядків;
* функції маніпулювання рядками.

Функції наводяться у вигляді списків, що згруповані за їх розташуванням у заголовних файлах.

Розглянемо найбільш уживані функції.

***Функції копіювання рядків:***

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

**char str [50];**

**strcpy (str, “О деле суди по исходу.”);**

* **char \*strdup (const char \*str);** — виконує копіювання рядка **str** і повертає покажчик на рядок-копію, наприклад:

**char st1[] = “Слово — есть поступок.”;**

**char \*st2;**

**st2 = strdup (st1);**           //копируется st1 в st2;

**Функції, конкатенації рядків:**

* **char \*strcat (char \*st1, const char \*st2);** — поєднує **st1 і st2** та повертає **st1**,наприклад:

**char str [100];  
strcpy (str, “Borland “);  
strcat (str, ” C++5″);**

у результаті маємо рядок

**string = “Borland C++5”;**

* **char \*strncat (char \*st1, const char \*st2, int n);** — додає до рядка **st1 n** символів рядка **st2** і повертає знову в **st1**, наприклад:

**char st1 [90] = “Привет “;**

**char st2 [50] = “студент и студентка”;**

**strncat (st1, st2, 7);,**

у результаті маємо рядок:

**st1 = “Привет студент ” .**

***Функції порівняння рядків:***

* **int strcmp (char \*stl, char \*st2);** — порівнює рядки **st1 і st2** та повертає цілу величину, що дорівнює:

**<0 — якщо st1 < st2;**

**= 0 — якщо st1 = st2;**

**>0 — якщо st1 > st2;**

наприклад:

**char st1[ ] = “Слово ” ;**

**char st2[ ] = “слово”;**

**int k;**

**k = strcmp (st1, st2);**        //k<0;

**int stricmp (const char \*stl, const char \*st2);** — виконує порівняння рядків, не враховуючи регістра символів; повертає цілу величину, як і функція **strcmp()**, наприклад:

**char st1[ ] = “Слово “;**

**char st2[ ] = “слово”;**

**int k;**

**k = stricmp (st1, st2);** //k = 0;

***Функції пошуку підрядка в рядку:***

* **int strspn (const char \*st1, const char \*st2 );** — повертає кількість символів від початку рядка **st1**, що збігаються із символами рядка **st2**, де б вони не знаходилися в **st2**,наприклад:

**char st1 [ ] = “Borland С++5”;**

**char st2 [ ] = ” narlBod “;**

**int k;**

**k= strspn (sti, st2);** — змінна **k**одержує значення, що дорівнює **8**, тому що перші 8 символів рядка містилися в **st1**(враховуючи символ пропуску);

* **char \*strstr (const char \*st1, const char \*st2);** — функція шукає в рядку **st1** перше входження **st2** і повертає покажчик на перший символ, знайдений у **st1**, з підрядка**st2**; якщо рядок **st2** не виявлений в **st1**, функція повертає **0**, наприклад:

**char stl [ ] = “Привет, сокурсник, идем на экзамен”;**

**char st2[ ] = “сокурсник”;**

**char spt;**

**spt = strstr (stl, st2);**

Результат виконання:

**spt = “сокурсник, идем на экзамен”.**

За потреби визначення останнього входження можна спочатку реверсувати рядок за допомогою функції **strrew;**

***Приклад 1.*** Скласти програму вилучення підрядка в **n** символів з **k**-ої позиції в рядку.

/\* *удаление подстроки в п символов из k-ой позиции в строке* \*/

**#include <iostream.h>**

**#include <string.h>**

**#include <conio.h>**

*//--------------- функция удаления подстроки из строки*

**int del (char\*sp, int k, int n)**

**{ int і;**

**for (і = k; і < strlen(sp); і++)**

**sp[i] = sp[i+n];**

**sp[i] = \0';}**

**main ()**           *//----------- главная функция*

**{ char st[50], pst[10];**

**cout << "\*\*\*\*\* Введите строку\n";**

**cin.getline(st, 50);**

**cout << "\*\*\*\*\* Введите подстроку\n";**

**сіn >> pst;**

**cout << "Исходная строка: — "<< st << endl;**

**del (st, strstr(st, pst)-st, strlen(pst));**

**cout << "Новая строка: — "<<st<<endl;**

**getch();**

**return 0;**

**}**

Результата обчислень:  
**\*\*\*\*\* Введите строку**  
**Люблю писать программы на языке С++!**  
**\*\*\*\*\* Введите подстроку**  
**писать**  
**Исходная строка: — Люблю писать программы на языке С++!**  
**Новая строка: — Люблю программы на языке С++!**

**Функції перетворення рядків у числа та чисел у рядки знаходяться у файлі stdlib.h:**

* **int atoi (const char \*s);** — перетворює рядок **s** у число типу **int**. Повертає отримане число **0**, якщо зустрінеться символ, що не може бути перетворений. Рядок повинен містити число, наприклад, **«2345»**, та мати таку структуру: **[пропуски] [знак числа] [цифри];**
* **long atol (const char \*s);** — перетворює рядок **s** у число типу **long** **int** **(**аналогічна функції **atoi.**);
* **double atof (const char \*s);** — перетворює рядок символів у число з плаваючою крапкою типу **double**. Якщо зустрічається символ, що не може бути перетворений, повертає **0**. Оброблюваний рядок повинен мати таку структуру: **[пропуски] [знак числа] [цифра.цифра] [літера е, Е, d або D] [знак порядку] [цифри порядку]**, наприклад, **«-12345.123» або «-12.345123 ЕЗ»**;

***Приклад 2.*** Ввести до пам’яті комп’ютера список прізвищ, які розташовані в будь-якому порядку, та відсортувати їх за алфавітом.  
Розглянемо перший варіант реалізації поставленої задачі. Будемо вважати, що вводять прізвища та ініціали, тоді програма може мати вигляд:

// *отсортировать фамилии по алфавиту*

**#include <iostream.h>**

**#include <string.h>**

**#include <conio.h>**

**main( )**

**{ const int n=5;**

**char sp[n][l5], r[15];**

**int i, k;**

*//------------------------------- ввод фамилий и инициалов*

**cout<< "\*\*\*\*\* Введите " << n << " фамилий \n";**

**for (і = 0; і < n; і++)**

**{ cout<<"Введите "<<(і+1)<<" фамилию и инициалы\n";**

**cin.getline (sp[i], sizeof (sp[i]) - 1);**

**}**

*//------------------------------ сортировка списка фамилий*

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

**for (i = 0; і < n-k; i++)**

**if (strcmp (sp[i], sp[i+l])>0)**

**{ strcpy (r, sp[i]);**

**strcpy (sp[i], sp[i+1]);**

**strcpy (sp[i+1], r);}**

**cout<<"\n Отсортированный массив фамилий \n";**

**for (і = 0; і < n; i++)**

**cout << sp[i] << endl;**

**getch ();**

**}**

Результати обчислень:  
**\*\*\*\*\*\*\* Введите 5 фамилий**  
**Введите 1 фамилию и инициалы**  
**Иванченко С. И.**  
**Введите 2 фамилию и инициалы**  
**Авдиенко А. Р.**  
**Введите 3 фамилию и инициалы**  
**Яшин Б. Ю.**  
**Введите 4 фамилию и инициалы**  
**Кашкин Т. Б.**  
**Введите 5 фамилию и инициалы**  
**Мельниченко Т. Ю.**  
**Отсортированный массив фамилий**  
**Авдиенко А. Р.**  
**Иванченко С. И.**  
**Кашкин Т. Б.**  
**Мельниченко Т. Ю.**  
**Яшин Б. Ю.**

У наведеній програмі використано масив прізвищ **sp [6][15]** і символьний рядок **r**, який потрібен для тимчасового зберігання прізвища при сортуванні масиву. Для сортування був метод виштовхування («пухирця»).

Порівняння елементів символьного масиву **(char sp[n][15])** здійснюється за допомогою функції **strcmp( )**, а перезапис прізвищ з одного елемента масиву **sp[i]** в другий — **sp[i+1]**— за допомогою функції **strcpy( )** і змінної **r**.

Після сортування на екран виведено одержаний масив.

Другий варіант розв’язання поставленої задачі використовує покажчики.

/\* *сортировка списка фамилий в алфавитном порядке с использованием указателей* \*/

**#include <iostream.h>**

**#include <string.h>**

**#include <conio.h>**

**void main( )**

**{ const n=5;**

**char sp [n][15];**

**int i, k;**

**char \*ps[n], \*ptr;** //*ps[n] — массив указателей*

*// ввод фамилий и инициализация массива указателей*

**cout << "\*\*\*\*\* Введите фамилии \n";**

**for (і = 0; і < n; і++)**

**{ gets (sp [і]);**

**ps[i] = sp[i];}**

*//--------------------------- вывод исходной информации*

**cout << "\n\*\*\*\*\* Исходный список\n";**

**for (і = 0; і < n; і++)**

**puts (ps[i]);**

//------------------------------------ сортировка массива

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

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

**if (strcmp (ps[i], ps[i+1]) > 0)**

**{ ptr = ps[i];**

**ps[i] = ps[i+1];**

**ps[i+1] = ptr;   }**

*//--------------------- вывод отсортированного массива*

**cout << "\n\n\*\*\*\*\*Отсортированный список \n";**

**for (і = 0; і < n; і++)**

**puts (ps[i]);**

**getch ();**

**}**

Результати виконання програми:  
**\*\*\*\*\* Введите фамилии**  
**Игнатенко А. P.**  
**Головко Н. А.**  
**Долбня Б. В.**  
**Андриенко С. Ф.**  
**Ичко Т. В.**  
**\*\*\*\*\* Исходный список**  
**Игнатеяко А. Р.**  
**Головко Н. А.**  
**Долбня Е. В.**  
**Андриенко С. Ф.**  
**Ичко Т. В.**  
**\*\*\*\*\* Отсортированный список**  
**Андриенко С. Ф.**  
**Головко Н. А.**  
**Долбня Е. В.**  
**Игнатенко А. Р.**  
**Ичко Т. В.**  
  
**Використання рядків типу string**

**Р**аніше розглядалися питання обробки символьних даних мови С++, у тому числі і символьних рядків. Однак в останніх версіях мови С++, починаючи з С++4.5, введена стандартна бібліотека шаблонів — **Standard Template Library (STL)**, яка містить клас **string** з більш потужними засобами обробки рядків.

Для підключення цього класу до програми слід записати директиву:

**#include <string>** (без розширення ***.h***)

і підключити простір імен бібліотеки шаблонів у вигляді

**using namespace std; .**

Після цього можна оголошувати змінні типу **string:**

**string strl, str2;.**

Ініціювання рядків при оголошенні виконується одним із способів:

**string st1 = “Це рядок класу string”;**  
**string st2 (“Це інший рядок класу string”);.**

Значення рядка **string** містить будь-який набір символів, записаний у лапках.

Для рядків типу **string** визначено такі операції:

* конкатенації (приєднання), котрі позначаються символом «+»;
* відношення («==», «!=», «>», «>=», «<», «<=»).

Наприклад, фрагмент  
**string st1 = “Приклад”;**  
**string st2 = “рядка”;**  
**string st3 = st1+’ ‘+st2;**  
**cout << st3 << endl;**  
дозволить вивести на екран повідомлення: **Приклад рядка.**

Для введення рядків **string**, крім операторів присвоювання, застосовують оператори введення даних:  
**сіn >> st;**  
**cin.getline(st, siseof(st));**  
**getline(cin.st, ‘\n’); тощо.**

Виведення рядків на екран здійснюється шляхом використання звичайних операторів виведення даних.

Рядки можна об’єднувати у масиви, які оголошуються звичайним засобом, тобто

**string sp[10];** — оголошення масиву, що містить 10 рядків.

Доступ до символів рядка здійснюється шляхом запису порядкового номера символу — індексу, який починається з нуля. Індекси можна записувати як у квадратних дужках, так і у звичайних, круглих.

Наприклад, якщо записати  
**string str=”Mій рядок”;**  
то **str[2]** — це буде літера **‘й’.**

Для масивів рядків потрібний символ визначається шляхом запису двох індексів: індексу елемента масиву та індексу символу в цьому елементі, тобто у вигляді **mas[i][j].**

**Функції для обробки рядків типу string**

Існує багато функцій для обробки рядків типу **string,** розглянемо деякі з них.

***Функції визначення довжини рядка:***

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

Наприклад:

*//--------------------------- определение длины строки*

**#include <iostream.h>**

**#include <string>**

**using namespace std;**

**main()**

**{ string str, st ="И вот сама пришла волшебница зима!";**

**str =" волшебница";**

**cout << "Длина строки str = " << str.size() << "Длина строки st = " << st.length() << endl;**

**}**

Результат виконання:  
**Длина строки str =11 Длина строки st = 34**

***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.**

Наприклад:

*//--------------- додавання рядка*

**string str, st = "Substring in text";**

**str.append(st,3,6);**

**cout << "str = "<<str << endl;**

Результат виконання:  
**str = string**

***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 підрядка в рядок*  
**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:  
// вилучення підрядка з рядка  
**string str = “My text string”;**  
**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:  
// *заміна підрядка в рядку*  
**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:  
*//————- обмін вмісту рядків*  
**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:  
*//————- виділення частини рядка*  
**string str, st;  
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:  
// *позиція входження підрядка в рядок*  
**string str, st=”nopa”;**  
**str=”Oceнняя пора – дождливая пора!”;**   
**int p=str.find(st,0);**  
**cout <<” p=”<<p<<endl;**

Результат виконання:  
**р=8.**  
**string str, st=”nopa”;**  
**str=”Oceнняя пора – дождливая пора!”;  
int p=str.rfind(st,str.length);  
cout <<” p=”<<p<<endl;**  
Результат виконання:  
**р=25.**

***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.**

**Приклади використання рядків типу string.**

***Приклад 4.*** Увести список прізвищ і відсортувати його за алфавітом

// *сортировка фамилий по алфавиту*

**#include <iostream.h>**

**#include <string>**

**#include <conio.h>**

**using namespace std;**

**main()**

**{ const n=5;**

**string spis[n];**

**int i, k;**

*//--------------------------- ввод списка фамилий*

**for(i = 0; і < n; i++)**

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

**getline(cin,spis[i],'\n');  }**

//--------------------------- сортировка списка фамилий

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

**for (і = 0; і < n-k; i++)**

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

*//--------------------------- вывод отсортированного списка*

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

**for(i = 0; і < n; i++)**

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

**getch ();**

**}**

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 5**

/// *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.**

***Контрольні запитання для самоперевірки***.

1. Що таке рядки та значення елементів символьного типу?
2. Що являє собою масив символьного типу?
3. Як здійснюється введення символьних даних?
4. Які операції можна здійснити з рядками типу string?
5. Як можна ініціювати рядки типу string?
6. Як можна визначити символ у рядках типу string?
7. Як здійснюється введення рядків типу string?
8. Як виконується порівняння даних рядків типу string?
9. Як визначити кількість символів у рядку типу string?
10. Які функції мови С++ необхідні для виділення підрядка з рядка типу string?

**Для самостійного вивчення** *(2 години)*: Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

**Рекомендована література**

1. Ковалюк Т. В. Алгоритмізація та програмування: Підручник. — Львів: «Магнолія 2006», 2013. — 400 с., ил.
2. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб.: Питер, 2003. – 461 с. URL: <http://www.ph4s.ru/bookprogramir_1.html>
3. Вступ до програмування мовою С++. Організація обчислень: навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставовський. – К. : Видавничо-поліграфічний центр "Київський університет", 2012. – 175 с. с.: іл. ISBN (укр.) . URL: <http://csc.knu.ua/uk/library/books/belov-24.pdf>
4. Джейс Либерти Освой самостоятельно С++ за 21 день: 3-е изд. пер. с англ.: Уч. пос. – М.: Издательский дом „Вильямс”, 2001. – 816 с.: ил..

5. Дейтел Х., Дейтел П. Основы программирования на С++. – М.: Бином, 1999. – 1024 с. URL: <http://ijevanlib.ysu.am/wp-content/uploads/2018/03/deytel.pdf>