**Лабораторна робота №4. Використання рядків типу string, символьних та рядкових величин.**

**Мета*:*** *навчитись складати й реалізовувати алгоритми та програми мовою С++ для обробки рядків типу string, символьних та рядкових величин, використовуючи оператори введення та виведення до файлу.*

***Завдання:***

1. Ознайомитися з теоретичними відомостями до цієї ЛР, переглянути теоретичні відомості до ЛР №3 та врахувати їх при виконанні завдання і в подальшій роботі.
2. Запустити середовище програмування С++
3. Продовжуємо працювати з програмою, розробленою на ЛР№3. Якщо ви обрали тему курсової роботи, то ЛР№3, 4, то виконуйте ці роботи відповідно до своєї теми.
4. У вікні редагування доповните текст програми, відповідно до наданої постановки задачі.
5. В першому рядку програми записати в коментарі номер групи та прізвище, а також номер ЛР, при виконанні ЛР за темою курсової роботи (КР) – додаєте коментар з назвою КР.
6. Результати надсилати на електронну адресу викладача

ЗАУВАЖЕННЯ

Приклад рядка, сформованого програмою

1;А.В. Ovchieva;1;11534336;121;3; #20:10:5

Цей рядок містить лише дані, відділені один від одного розділювачами.

[**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді cpp-файлу з іменем у форматі

**<Номер групи> <Номер лабораторної><Прізвище англійською>**

Наприклад, 31-01Ivanov.cpp.

Тему в заголовку листа записати

**ООП<Номер групи>-ЛР <Номер лабораторної>-<Прізвищеанглійською>**

**Строк відсилки ЛР ІПЗ-31 25.11.2023**

**ІПЗ-32 18.11.2023**

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

**ООП<Номер групи>-Запитання-<Прізвище англійською>**.

**Виконати тест (поточне опитування – ООП Тест 1)**

**https://docs.google.com/forms/d/e/1FAIpQLSc-AQu5F6Ry7JQEKXCzfEhXpSQnpRdOIPxwbutx6P1AI\_tmOw/viewform?vc=0&c=0&w=1&flr=0**

**Строк виконання ІПЗ-31 17.11.2023**

**ІПЗ-32 14.11.2023**

**Постановка задачі:**

1. Продовжуємо розроблювати додаток, який буде оброблювати інформацію про викладачів та студентів.
2. Додати позиції меню "reading from the teacher data file " (читання з файлу даних про викладачів) (4), "reading from a data file about students " (читання з файлу даних про студентів) (5), " sorting teacher data by phone number " (сортування за номером телефону даних про викладачів) (6), "sorting student data by phone number " (сортування за номером телефону даних про студентів) (7).
3. Позиції меню (4) та (5) реалізуються аналогічно: виконується читання даних зі сформованого відповідного файлу і заповнюється відповідний масив структур.
4. Позиції меню (6) та (7) реалізуються аналогічно: виконується сортування інформації в масиві структур, після чого виконується запит щодо виведення відсортованої інформації на консоль і при запиті на виведення вона виводиться. Потім робиться запит щодо перезапису файлу і при запиті га перезапис він перезаписується.
5. Дії в позиціях меню оформлюються викликом відповідної функції, де вони реалізуються.
6. Результат виконання ЛР: модифікований файл програми (срр) та два відсортованих масиви.

**Теоретичні відомості.**

1. **Щодо виклику main() з main()** в <https://ru.cppreference.com/w/cpp/language/main_function> надаються такі відомості:

# Функция main()

Программа должна иметь глобальную функцию **main**, которая является точкой старта программы. Она должна иметь одну из следующих форм:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| int **main** **()** **{** body **}** | (1) |  |  |  |  |  |  |  |  |
| int **main** **(**int argc**,** char \*argv[]**)** **{** body **}** | (2) |  |  |  |  |  |  |  |  |
| /\* другая, определённая реализацией форма, с int, в качестве возвращаемого типа \*/ | (3) |  |  |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| argc | — | Неотрицательное число, представляющее количество аргументов, переданных программе из окружения, в котором запустили программу. |
| argv | — | Указатель на первый элемент массива argc + 1 указателей, последний из которых равен null, а предыдущие, если существуют, указывают на [многобайтовые строки с завершающим нулём](https://ru.cppreference.com/w/cpp/string/multibyte), которые представляют аргументы, переданные программе из окружения, в котором её запустили. Если argv[0] не пустой указатель (или, что тоже самое, argc > 0), он указывает на строку, которая представляет имя, использованное при запуске программы, или пустую строку. |
| body | — | Тело функции main. |

Имена argc и argv могут быть произвольными, так же как и запись типов этих параметров: int main(int ac, char\*\* av) так же допустимо.

Самая распространённая форма main(), определённая реализацией, имеет третий аргумент (дополнительно к argc и argv), типа char\*[], указывающий на [на массив указателей на переменные среды выполнения](http://pubs.opengroup.org/onlinepubs/9699919799/functions/exec.html).

### Объяснение

Функция main вызывается при старте программы после [инициализации](https://ru.cppreference.com/w/cpp/language/initialization) нелокальных объектов со статическим [классом памяти](https://ru.cppreference.com/w/cpp/language/storage_duration). Это точка входа в программу, которая исполняется в *гостевом* окружении (то есть с операционной системой). Точки входа в *автономные* программы (загрузчики, ядра ОС, и т.п.) зависят от реализации.

Параметры функции main в варианте с двумя параметрами позволяют передать произвольные многобайтовые строки из среды выполнения (обычно известных как *аргументы командной строки*), указатели argv[1] .. argv[argc-1] ссылаются на первые символы этих строк. argv[0] это указатель на первый символ многобайтовой строки с завершающим нулём, которая содержит имя, используемое при вызове программы (или пустую строку "", если это не поддерживается средой выполнения). Эти строки изменяемые, хотя их изменения не распространяются назад в среду выполнения: они могут использоваться, например, в [std::strtok](https://ru.cppreference.com/w/cpp/string/byte/strtok). Размер массива, на который указывает argv, равен, по меньшей мере, argc+1, и последний элемент массива argv[argc] гарантированно является пустым указателем.

Функция main обладает следующими специальными свойствами:

1) Она нигде не может быть использована в программе,

а) в частности её нельзя вызывать рекурсивно;

б) нельзя взять её адрес.

2) Её нельзя объявлять и нельзя перегружать: фактически имя main зарезервировано в глобальном пространстве имён для функций (хотя это имя может быть использовано для именования классов, пространств имён, перечислений и любых сущностей не в глобальных пространствах имён, за исключением того, что функция с именем 'main' не может быть объявлена со [связыванием](https://ru.cppreference.com/mwiki/index.php?title=cpp/language/%D1%81%D0%B2%D1%8F%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC&action=edit&redlink=1) для языка C в любом пространстве имён (начиная с C++17)).

3) Её нельзя объявить как удалённую или определить со [связыванием](https://ru.cppreference.com/mwiki/index.php?title=cpp/language/%D1%81%D0%B2%D1%8F%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC&action=edit&redlink=1) для языка C (начиная с C++17), [inline](https://ru.cppreference.com/w/cpp/language/inline), [static](https://ru.cppreference.com/w/cpp/language/static) или [constexpr](https://ru.cppreference.com/w/cpp/language/constexpr).

4) В теле функции main не обязателен [оператор return](https://ru.cppreference.com/w/cpp/language/return): при завершении функции main без оператора return эффект будет тот же самый, как при выполнении return 0;.

5) Выполнение return (или неявного return при достижении конца функции main) эквивалентно нормальному выходу из функции (которое уничтожает объекты с автоматическим классом памяти) с последующим вызовом [std::exit](http://ru.cppreference.com/w/cpp/utility/program/exit) с тем же самым аргументом, который был передан в [return](https://ru.cppreference.com/w/cpp/language/return). ([std::exit](http://ru.cppreference.com/w/cpp/utility/program/exit) уничтожает статические объекты и завершает программу).

6) (начиная с C++14) Тип возвращаемого значения функцией main не может быть выведен (auto main() {...} не разрешено).

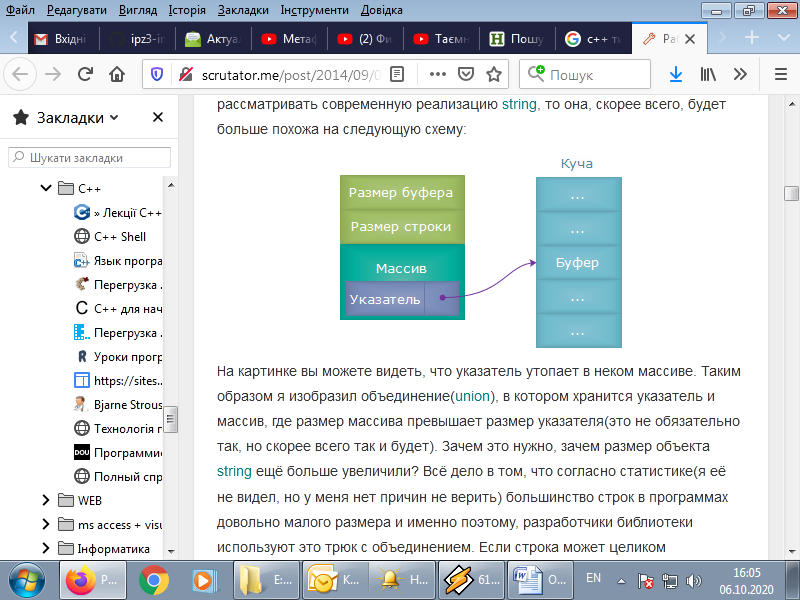
Окрім того стандартом мови забороняється будь-яке використання функції main:

**6.6.1 main function [basic.start.main]**  
3 The function main shall not be used within a program.

1. **Спосіб збереження рядка string**

За матеріалами <http://scrutator.me/post/2014/09/02/cpp_strings_basics.aspx> (у мене перестала працювати)

В статті розглядається спосіб збереження рядка **string**



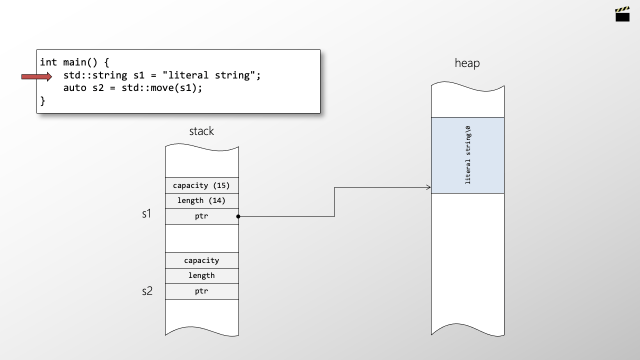
Масив – це позначене місце зберігання даних класу **string**, якщо кількість символів не перевищує розміру масиву. В іншому випадку в "купі" виділяється буфер, розмір, якого більший за дійсний розмір рядка, і покажчик вказує адресу цього буферу.

Ще про збереження рядка string

# Урок №210. Довжина і ємність std::string

<https://acode.com.ua/urok-210-dovzhyna-i-yemnist-std-string/>

# Для більш детального знайомства зі зберіганням string в пам‘яті доцільно звернутися до <https://blog.feabhas.com/2022/02/working-with-strings-in-embedded-c/> (Working with Strings in Embedded C++) Нижче фрагмент з цього тексту



## *Масиви рядків*

Текст, що обробляється програмою, може розглядатися як масив рядків. У такому використанні немає нічого незвичайного. Наприклад, при програмуванні баз даних для з'ясування коректності команд, що вводяться користувачем, вхідні дані порівнюються з вмістом масиву рядків, у якому записано допустимі у цьому додатку команди.

Для побудови масиву рядків може використовуватися двовимірний символьний масив, у якому розмір лівого індексу визначає кількість рядків, а розмір правого – максимальну довжину кожного рядка. Наприклад, char strArray[20][80] – оголошення масиву, призначеного для зберігання 20 рядків завдовжки 80 символів.

Отримати доступ до окремого рядка такого масиву достатньо просто: достатньо вказати тільки лівий індекс. Наприклад, виклик функції gets для запису третього рядка описаного вище масиву:

gets(strArray[2]);

Щоб краще зрозуміти, як потрібно поводитися з масивами рядків, розглянемо фрагмент програми, яка приймає рядки тексту, що вводяться з клавіатури, і відображає їх на екрані після введення порожнього рядка.

int t, i;

char text[100][80];

for(t=0; t<100; t++)

{ cout << t << ": "; gets(text[t]);

if(!text[t][0]) break; *// вихід з циклу за порожнім рядком*

}

for(i=0; i<t; i++) *// відображення рядків на екрані*

cout << text[i] << endl;

Зверніть увагу на те, як у програмі здійснюється перевірка на введення порожнього рядка. Функція gets повертає рядок нульової довжини, якщо єдиною натиснутою клавішею виявилася клавіша Enter. Це означає, що першим байтом у рядку буде нуль-символ. Нульове значення завжди інтерпретується як помилкове, але узяте із запереченням (!) – воно дає значення *істина*, яке дає змогу виконати негайний вихід з циклу за допомогою оператора break.

До кожного окремого рядка масиву рядків можна також звернутися за допомогою покажчика на його перший символ. Такий варіант зручний тим, що для перестановки двох рядків фактично достатньо переставити покажчики в масиві покажчиків, а не самі рядки.

Розглянемо функцію month\_name, яка повертає покажчик на рядок, що містить ім'я n-го місяця.

#include <iostream.h>

#define n 14

char \*month\_name (int ); *// назва k-го місяця*

void main ()

{ char \*month\_name(int);

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

cout << "month number " << i << " - " << month\_name(i) << endl;

}

char \*month\_name (int k)

{ static char \*name[] = {"incorrect month","January", "February","March","April", "May", "June", "July", "August", "September", "October", "November", "December"};

return (k<1||k>12)?name[0]: name[k];

}

Функція month\_name містить локальний масив рядків і при зверненні до неї повертає покажчик на потрібний рядок.

В описі масиву покажчиків на символи name[] ініціалізатором є просто список рядків. Символи i-го рядка поміщаються в певне місце пам'яті, а покажчик на його початок зберігається в елементі name[i]. Оскільки розмір масиву name не зазначений, компілятор сам підраховує кількість ініціалізаторів і відповідно встановлює правильне число.

Використання покажчиків при роботі із текстовою інформацією може бути значно складнішим. Наприклад, розглянемо наступну програму:

char \*c[]={"ENTER","NEW","POINT","FIRST"};

char \*\*cp[]= {c+3,c+2,c+1,c};

char \*\*\*cpp=cp;

void main ()

{ cout << \*\*++cpp;

cout << \*--\*++cpp+3 << " ";

cout << \*cpp[-2]+3;

cout << cpp[-1][-1]+1 << endl;

}

Описана структура схематично представлена на рис. 2.

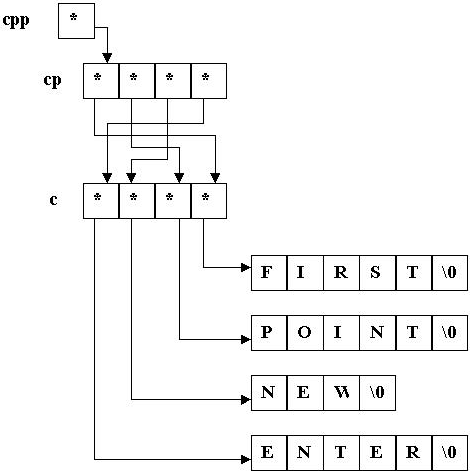


Рис. 2. Вихідна структура даних

Тут c - масив покажчиків на символ. Його елементи ініціалізуються так, що вони вказують на масиви символів "ENTER", "NEW", "POINT" і "FIRST". cp – масив покажчиків на покажчики на рядки. Його елементи ініціалізуються так, що вони вказують на елементи масиву c. Оголошення \*\*cp[] відповідає символу, \*cp - покажчику на символ і cp[] - покажчику на покажчик на символ. Нарешті, cpp - покажчик, що вказує на покажчик на покажчик на символ. Оголошення \*\*\*cpp відповідає символу, \*\*cpp – покажчику на символ, \*cpp - покажчику на покажчик на символ.

Виконання оператора \*\*++cpp або що теж саме \*(\*(++cpp)) збільшує cpp наступним чином (рис. 3):

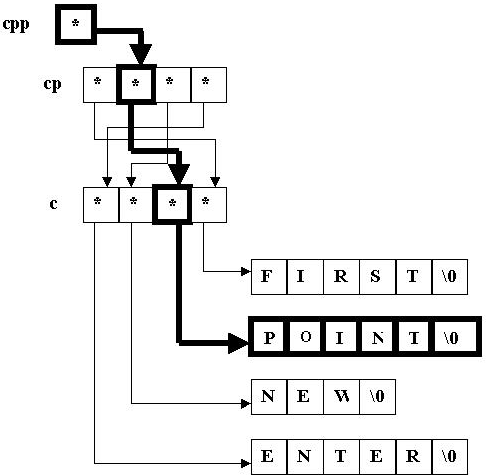


Рис. 3. Схема обчислення виразу \*\*++cpp

Виконання оператора \*--\*++cpp+3 або що теж саме (\*(--(\*(++cpp))))+3 збільшує cpp, за покажчиком доходить до cp[2], зменшує cp[2], за покажчиком доходить до c[0] і цей покажчик (тобто c[0]) збільшує на 3 (рис. 4).

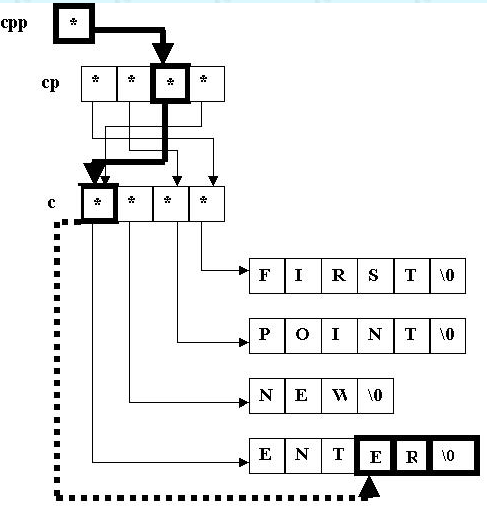


Рис. 4. Схема обчислення виразу \*--\*++cpp+3

Виконання оператора \*cpp[-2]+3 або що теж саме (\*(cpp[(-2)]))+3 індексує cpp значенням -2. Отримаємо cp[0], за покажчиком дійдемо до c[3] і проіндексуємо його значенням 3 (рис.5)

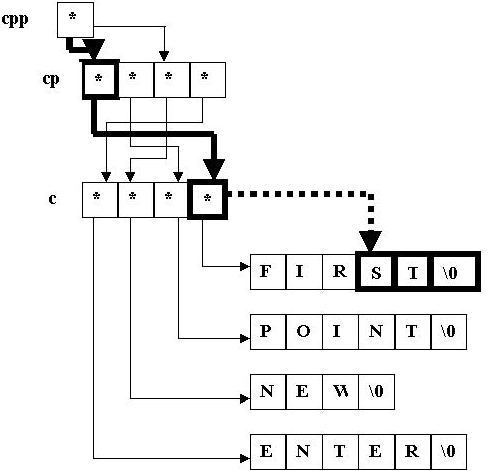


Рис. 5. Схема обчислення виразу \*cpp[-2]+3

При виконанні оператора cpp[-1][-1]+1 або що теж саме ((cpp[-1])[-1])+1, індексуючи cpp значенням -1, отримаємо cp[1], знову індексуємо з -1 і доходимо до c[1], це останнє посилання індексуємо значенням 1 (рис.6)

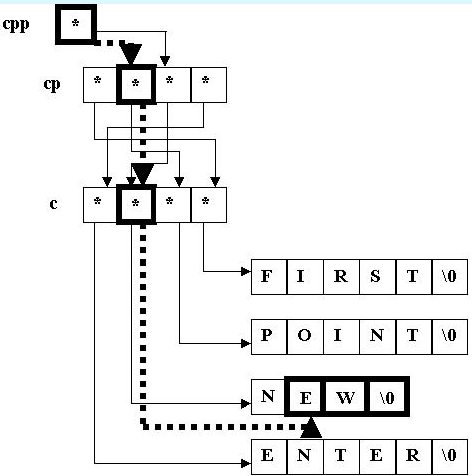


Рис. 6. Схема обчислення виразу cpp[-1][-1]+1

Приклад. Замінити у заданому рядку перше входження заданого слова вказаним новим словом.

#include <stdlib.h>

#include <stdio.h>

#include < string.h>

#include <iostream>

using namespace std;

char \*s1 = "I see nothing.";  *// вихідний рядок*

char \*a = "see", *// слово, яке треба замінити*

\*b = "hear"; *// нове слово*

char s2 [100]; *//допоміжний рядок*

int main()

{ strcpy(s2, s1); *// скопіювати вихідний рядок у допоміжний*

char \*p1 = strstr(s1, а); *// встановити p1 на слово а в оригіналі*

char \*р2 = strstr(s2, а); *// встановить р2 на слово а в копії*

strcpy(p2, b); *// копіювати слово b*

p1 += strlen(a); *// змістити покажчик в оригіналі*

р2 + = strlen(b); *// змістити покажчик в копії*

strcpy(р2, р1); *// скопіювати залишок рядка*

cout << s2 << endl ;

system("pause");

}