**Лабораторна робота №10. Написання програм з використанням рядкових величин та структур**

**На період** карантину в дистанційній формі навчання потрібно використовувати замість **С++ Builder 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-2110P**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 англійською>**

**Строк відсилки ЛР 14.04.2021.**

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

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

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

**Методичні вказівки щодо організації самостійної роботи студентів**

1. Прочитати теоретичні відомості, повторити Лекцію 9, відповісти на контрольні запитання для самоперевірки.
2. Запустити середовище програмування С++ .
3. Записати програму, що виконує завдання з пп.4, яке складається з 2-х частин. В першому рядку програми записати

*// ПТБД-21 Група № ЛР№ Прізвище*

вказавши номер своєї групи, номер ЛР та своє прізвище.

Як і в ЛР №5, програма повинна запитати номер завдання (число 1, 2 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. Скласти програму оброблення рядків типу string**.**

1. 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. Знайдіть у першому реченні позицію слова «than» і поставте перед ним слово «learn», не забувши розділити їх пробілом.
3. Об‘єднайте отриманий результат з другим реченням і запишіть результат до третього рядка, знайдіть його довжину і виведіть як перші два рядка.

Додаткові бали за виконання цього ж завдання **ще і у варіанті**, коли рядки розглядаються як символьні масиви (без використання рядків типу string).

2. Скласти програму формування показників успішності студентів, використовуючи змінні типу структура.

1. Ввести в комп’ютер відомість успішності студентів, які здали сесію з дисциплін: «Математика», «Фізика» і «Програмування», та обчислити:

* середній бал кожного студента;
* середній бал групи за кожним предметом;
* вивести на екран прізвища відмінників з математики.

1. Розробіть структуру, що містить відомості про результати екзаменів по предмету:  
   Прізвище та ініціали,  
   Число балів по першому предмету,  
   Число балів по другому предмету,  
   Число балів по третьому предмету.
2. Ввести інформацію по студентах с консолі, попередньо зазначивши їх кількість.
3. Розрахуйте середній результат по групі, визначте відмінників з математики і виведіть результати: введена інформація про студента, середні результати по групі по кожному предмету, список відмінників з математики.

**Контрольні запитання.**

1. Які оператори введення-виведення символьних масивів ви знаєте? В яких випадках вони використовуються?
2. Як визначається кінець рядка?

**2.** Як описуються дані типу структура?

**3.** Які типи полів може містити структура?

**4.** Які існують способи ініціювання полів структури?

**5.** Що таке «вкладеність» структур?

**6.** Як здійснюється робота з типом struct з використанням по­кажчиків?

**7.** Як реалізується доступ до елементів масиву типу структура?

**Теоретичні відомості.**

**Деякі зауваження щодо використання рядків типу string**

* 1. Числа, які присвоюються, тип string обробляє як текст, а не як числа. А це означає, що ними не можна маніпулювати як звичайними числами (наприклад, ви не зможете виконати з ними **арифметичні операції**). C++ автоматично НЕ конвертує їх в значення цілочисельних типів або в типи з плаваючою крапкою.
  2. ***Введення рядків***

Розглянемо наступний приклад:

**#include <iostream>**

**#include <string>**

**using namespace std; *// простір імен***

**int main()**

**{ string myName;**

**string myAge;**

**cout << "Enter your full name: ";**

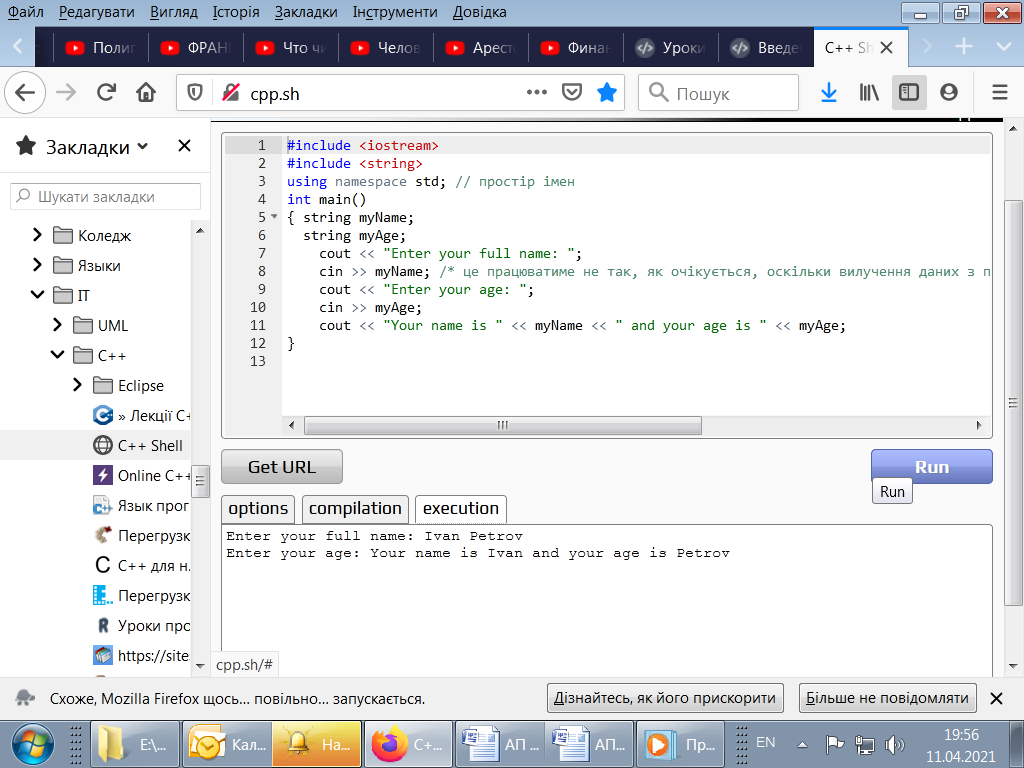
**cin >> myName; /\* це працюватиме не так, як очікується, оскільки вилучення даних з потоку std::cin зупиниться на першому пробілі\*/**

**cout << "Enter your age: ";**

**cin >> myAge;**

**cout << "Your name is " << myName << " and your age is " << myAge;**

**}**



Після натискання Enter одразу виводиться другий рядок. Причина: оператор виведення (>>) повертає символи з вхідного потоку даних тільки до першого пробілу. Всі інші символи залишаються всередині std::cin, очікуючи наступного витягування з потоку.

Тому, коли ми використовували оператор >> для вилучення даних в змінну myName, тільки Ivan вдалося витягнути, Petrov залишився всередині std::cin, очікуючи наступного витягування. Коли ми використовували оператор >> знову, щоб витягти дані в змінну myAge, ми отримали Petrov замість віку.

Щоб витягнути цілий рядок з вхідного потоку даних (разом з пробілами), використовуйте **функцію std::getline()**. Вона приймає два параметри: перший — std::cin, другий — змінна типу string. Потрібні нам оператори будуть мати вигляд:

getline(cin, myName);

getline(cin, myAge);

* 1. Витягування даних з std::cin за допомогою getline() іноді може призводити до несподіваних результатів. Наприклад, розглянемо наступну програму:

#include <iostream>

#include <string>

using namespace std; // простір імен

int main()

{string myName;

int choice;

cout << "Pick 1 or 2: ";

cin >> choice;

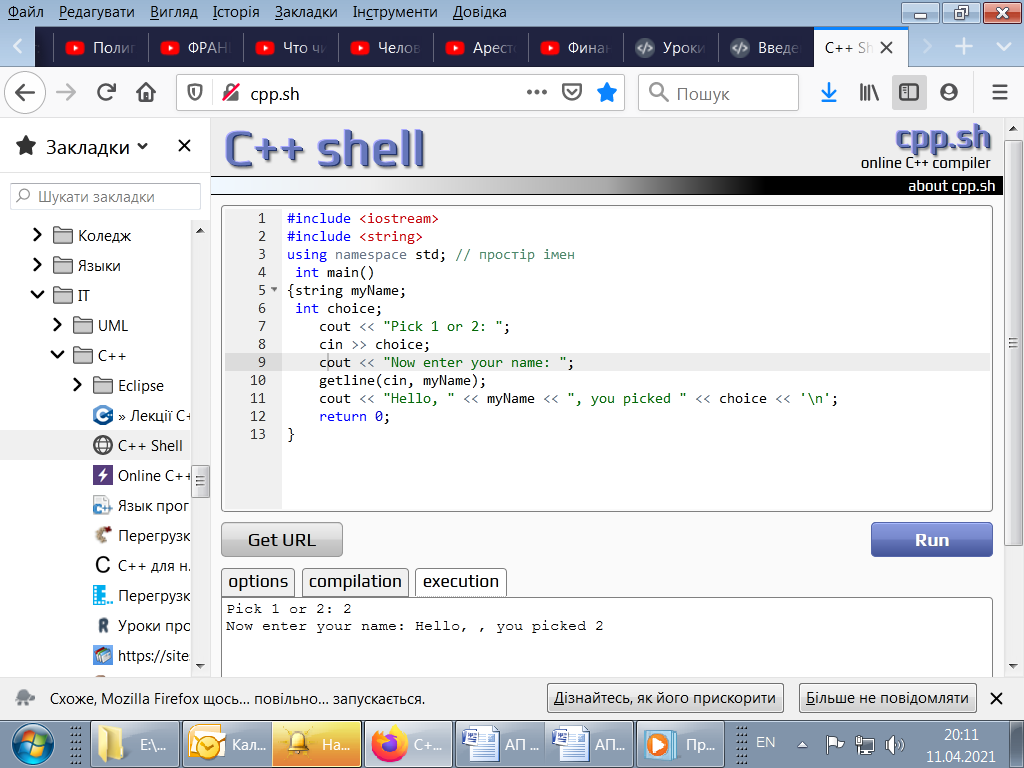
cout << "Now enter your name: ";

getline(cin, myName);

cout << "Hello, " << myName << ", you picked " << choice << '\n';

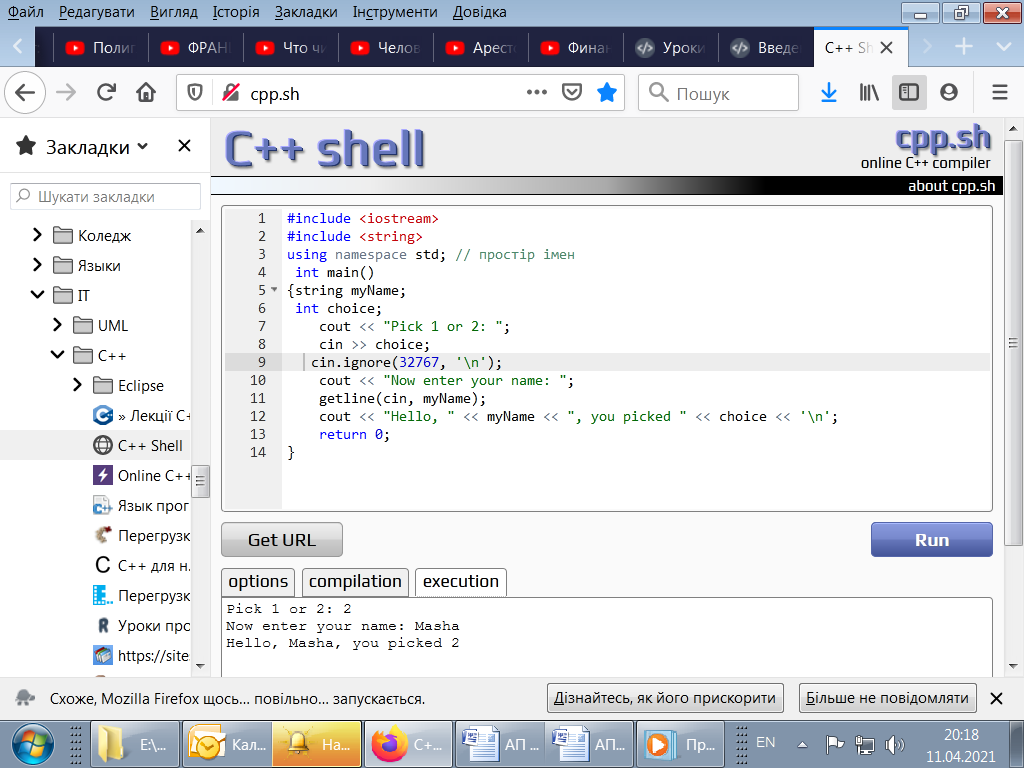
return 0;

}



Коли ви запустите цю програму, і вона попросить вас ввести ваше ім’я, вона не чекатиме вашого введення, а відразу виведе результат (просто пробіл замість вашого імені)! Виявляється, коли ви вводите числове значення, потік cin захоплює разом з вашим числом і символ нового рядка. Тому, коли ми ввели 2, cin фактично отримав 2\n. Потім він витягнув значення 2 в змінну, залишаючи \n (символ нового рядка) у вхідному потоці. Потім, коли getline() отримує дані для myName, він бачить в потоці \n і думає, що ми, мабуть, ввели просто порожній рядок! А це, безумовно, не те, що ми хочемо. **Хорошою практикою вважається видаляти з вхідного потоку даних символ нового рядка**. Це можна зробити наступним чином:

**cin.ignore(32767, '\n');**

****

* 1. Сучасний C++ підтримує **два різних типи рядків**:

std::string (як частина Стандартної бібліотеки мови С++);

рядки C-style (успадковані від мови Cі).

std::string реалізований за допомогою рядків C-style.

**Рядок C-style** — це простий **масив** символів, який використовує нуль-термінатор. **Нуль-термінатор** — це спеціальний символ (**ASCII-код** якого дорівнює 0), який використовується для позначення кінця рядку.

Для визначення рядку C-style необхідно просто оголосити масив типу char і ініціалізувати його **літералом** (наприклад, string):

**char mystring[] = "string";**

Хоча string має лише 6 букв, мова C++ автоматично додає нуль-термінатор в кінець рядку (нам не потрібно додавати його вручну). Відповідно, довжина масиву mystring дорівнює 7.

Рядки C-style слідують всім тим же правилам, що і масиви. Це означає, що ви можете ініціалізувати рядок при створенні, але після цього не зможете присвоювати йому значення за допомогою оператора присвоювання:

**char mystring[] = "string"; // ок**

**mystring = "cat"; // не ок!**

Це те ж саме, якби ми зробили наступне:

**int array[] = { 4, 6, 8, 2 }; // ок**

**array = 7; // що це значить?**

Оскільки рядки C-style є масивами, то ви можете використовувати оператор [] для зміни окремих символів в рядку:

**#include <iostream>**

**int main()**

**{**

**char mystring[] = "string";**

**mystring[1] = 'p';**

**std::cout << mystring;**

**return 0;**

**}**

Результат виконання програми:

spring

При виведенні рядка C-style, std::cout виводить символи до тих пір, поки не зустріне нуль-термінатор. Якби ви випадково перезаписали нуль-термінатор в кінці рядка (наприклад, присвоївши що-небудь для mystring[6]), то ви б не тільки отримали всі символи рядка, але std::cout також вивів би все, що знаходиться в сусідніх комірках пам’яті до тих пір, поки йому не попався би 0!

Зверніть увагу, що це нормально, якщо довжина масиву більше рядка, якого він зберігає:

**#include <iostream>**

**int main()**

**{**

**char name[15] = "Max"; /\* використовується тільки 4 символи (3 букви + нуль-термінатор)\*/**

**std::cout << "My name is: " << name << '\n';**

**return 0;**

**}**

В цьому випадку рядок Max виведеться на екран, а std::cout зупиниться на нуль-термінаторі. Решта символів в масиві будуть проігноровані.

Є багато випадків, коли ми не знаємо заздалегідь, наскільки довгим буде наш рядок. Наприклад, розглянемо проблему написання програми, де ми просимо користувача ввести своє ім’я. Наскільки довгим воно буде невідомо доти, доки користувач його не введе. У такому випадку ми можемо оголосити масив розміром більше, ніж нам потрібно:

**#include <iostream>**

**int main()**

**{**

**char name[255]; /\* оголошуємо достатньо великий масив (для зберігання 255 символів)\*/**

**std::cout << "Enter your name: ";**

**std::cin >> name;**

**std::cout << "You entered: " << name << '\n';**

**return 0;**

**}**

У вищенаведеній програмі ми оголосили масив з 255 символів, припускаючи, що користувач не введе ім’я більше 255 символів. Хоча це і поширена практика, але вона не дуже ефективна, так як користувачу нічого не заважає ввести ім’я більше 255 символів (випадково або навмисно). Набагато краще зробити наступне:

#include <iostream>

**int main()**

**{**

**char name[255]; /\* оголошуємо достатньо великий масив (для зберігання 255 символів)\*/**

**std::cout << "Enter your name: ";**

**std::cin.getline(name, 255);**

**std::cout << "You entered: " << name << '\n';**

**return 0;**

**}**

Виклик cin.getline() буде приймати до 254 символів в масив name (залишаючи місце для нуль-термінатора!). Будь-які зайві символи будуть проігноровані. Таким чином, ми можемо гарантувати, що масив не буде переповнений!

**Декілька найбільш корисних функцій:**

**Функція strcpy\_s()** — копіює вміст одного рядка в інший. Найчастіше це використовується для присвоювання значень рядку.

Використання strcpy\_s() може легко викликати **переповнення** масиву, якщо не бути обережним! У наступній програмі, довжина масиву dest менше довжини рядка, який ми копіюємо, тому в результаті ми отримаємо переповнення масиву:

**#include <iostream>**

**#include <cstring>**

**int main()**

**{**

**char text[] = "Print this!";**

**char dest[5]; /\* зверніть увагу, що довжина масиву dest всього 5 символів!\*/**

**strcpy\_s(dest, text); // переповнення!**

**std::cout << dest;**

**return 0;**

**}**

**Функція strlen()** — повертає довжину рядка C-style (не враховуючи нуль-термінатор):

**#include <iostream>**

**#include <cstring>**

**int main()**

**{**

**char name[15] = "Max"; /\* використовується тільки 4 символи (3 букви + нуль-термінатор)\*/**

**std::cout << "My name is " << name << '\n';**

**std::cout << name << " has " << strlen(name) << " letters.\n";**

**std::cout << name << " has " << sizeof(name) << " characters in the array.\n";**

**return 0;**

**}**

Результат виконання програми:

My name is Max

Max has 3 letters.

Max has 15 characters in the array.

Зверніть увагу на різницю між strlen() і sizeof. Функція strlen() виводить кількість символів ДО нуль-термінатора, тоді як [**оператор sizeof**](https://acode.com.ua/urok-44-umovnyj-ternarnyj-operator-sizeof-i-koma/#toc-0) повертає розмір цілого масиву, незалежно від того, що в ньому знаходиться.

**Функція strcat()** – додає один рядок до іншого (небезпечно).

**Функція strncat()** – додає один рядок до іншого (з перевіркою розміру місця призначення).

**Функція strcmp()** – порівнює два рядки (повертає 0, якщо вони рівні).

**Функція strncmp()** – порівнює два рядки до певної кількості символів (повертає 0, якщо вони рівні).

**Знати про рядки C-style варто, так як вони використовуються не так вже й рідко, але використовувати їх без вагомої на те причини — не рекомендується. Замість рядків C-style використовуйте std::string (підключаючи заголовок string), так як він простіший, безпечніший і гнучкіший.**