Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

**Лабораторная работа**

На тему:

" Строки типа string. Фильтрация данных. Vector “

**Выполнил:**

Волковец Сергей Николаевич

(подпись)

**Научный руководитель:**

Белодед Николай Иванович

(подпись)

Оглавление

[Теоритические сведения: 3](#_Toc192028924)

[Строка(String) 3](#_Toc192028925)

[Фильтрация данных 5](#_Toc192028926)

[Вектор 7](#_Toc192028927)

[Задания: 8](#_Toc192028928)

# Теоритические сведения:

## Строка(String)

Для хранения строк в C++ часто используют тип std::string из модуля <string>. Строки типа string имеют ряд преимуществ:

* возможность обработки строк стандартными операторами C++ ( =, +, = =, <> и т.п.);
* обеспечение лучшей надежности (безопасности) программного кода;
* возможность использования методов класса string для работы со строками;
* автоматическое управление памятью;
* возможность использования строки, как самостоятельного типа данных.

Для использования строк типа string в C++ необходимо подключить заголовочный файл string.

|  |
| --- |
| #include <string>  #include <iostream>  int main() {  std::string str = "Hello, world!";  std::cout << str << std::endl;  return 0;  } |

Ввод и вывод строк типа string можно осуществлять стандартными потоками ввода-вывода cin и cout.

|  |
| --- |
| string str; cin >> str; //hello world cout << str; //hello |

Следует обратить внимание на то, что строка читается до первого пробела.  
 Альтернативный способ ввода строки - использование функции **getline().** Она имеет следующий прототип:

|  |
| --- |
| istream& getline(istream& is, string& str, char delim); |

где:  
- is - входной поток, из которого необходимо считать строку;  
- str - объект типа `string`, в который нужно записать считанную строку;  
- delim - опциональный параметр, который указывает на символ-разделитель, до которого нужно считывать строку. По умолчанию разделитель ставится на символ переноса строки.

В данном примере, getline считывает всю строку до символа новой строки из потока ввода cin. Введенная строка сохраняется в переменной str.

|  |
| --- |
| string str; getline(cin, str); //hello world cout << str; //hello World |

Метод cin.ignore() предназначен для извлечения символов из входного потока (cin). Извлеченные символы игнорируются и не используются. Метод cin.ignore() может принимать один или два аргумента. Если метод вызывается с одним аргументом, то он игнорирует один символ. Если метод вызывается с двумя аргументами, то он игнорирует до n символов или до тех пор, пока не встретит символ разделитель (второй параметр).  
 cin.ignore(); игнорирует один символ.  
 cin.ignore(10); игнорирует до 10 символов.  
 cin.ignore(10,' '); игнорирует до 10 символов или или до тех пор, пока не будет достигнут пробел.

Пример использования:

|  |
| --- |
| string str; cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Очистка входного потока getline(cin, str);//input till '\n'  //hello world cout << str; //hello world |

Существует ряд удобных методов класса string для работы со строками:

* append() -объединение строк;

|  |
| --- |
| std::string str = "Hello"; str.append(", world!"); cout << str;//Hello, world! |

* assign() - присваивание строки;

|  |
| --- |
| std::string str1 = "Hello"; std::string str2 = "world"; str1.assign(str2); cout << str1;// str1 = "world" |

* insert() - вставка строки с заданной позиции;

|  |
| --- |
| std::string str = "Hello"; str.insert(5, ", world!"); cout << str;// "Hello, world!" |

* replace() - вызывающая строка заменяется другой строкой. Есть возможность задать позицию и количество символов в вызывающей строке, которые нужно заменить другой строкой;

|  |
| --- |
| std::string str = "Hello, world!"; str.replace(5, 6, "C++"); cout << str; // HelloC++d! |

* erase() - удаление указанного количества символов строки начиная с указанной позиции;

|  |
| --- |
| std::string str = "Hello, world!"; str.erase(5, 7); cout << str; // Hello! |

* find() - поиск подстроки в строке, возвращает индекс первого вхождения подстроки в строку. Если подстрока не найдена, то метод возвращает -1;

|  |
| --- |
| std::string str = "Hello, world!"; int pos = str.find("world"); cout << pos;// 7 |

* **string::npos** -это статическая константа в классе std::string, которая используется для обозначения конца строки. Она имеет значение -1, что означает, что она может быть использована для проверки наличия символа в строке. Если метод find() не находит символ в строке, он возвращает значение string::npos

|  |
| --- |
| std::string str = "Hello, world!"; int pos = str.find("c++"); cout << pos; // -1 |

* size() - получение размера строки;

|  |
| --- |
| std::string str = "Hello, world!"; int size = str.size(); cout << size;// 13 |

* empty() - проверка на пустоту.

|  |
| --- |
| std::string str = ""; bool is\_empty = str.empty(); cout << is\_empty;// 1 |

## Фильтрация данных

это процесс выборки данных из набора данных на основе определенных критериев. Фильтрацию можно производить по наличию определенной подстроки в строке.  
 Задача поиска подстроки в строке реализуется очень просто, если использовать встроенные функции для работы со строками типа string. Например, вот пример кода для поиска позиции вхождения подстроки “pa” в строке:

|  |
| --- |
| #include <iostream> #include <string>  using namespace std; int main() {  string s = "parapapa";  int i = 0;  for (i = s.find("pa", i++); i != string::npos; i = s.find("pa", i + 1))  cout << i << " "; //0 4 6  } |

Реализация этой же задачи, но при помощи массива типа char будет гораздо длиннее, но алгоритм все же достаточно прост:  
 Есть две строки. Например "Hello world" и "lo". Работать будем в два цикла:  
 1. Первый будет выполнять проход по всей строке, и искать местоположение **первой буквы** искомой строки ( "lo" ).  
 2. Второй, начиная с найденной позиции первой буквы – сверять, какие буквы стоят после неё и сколько из них подряд совпадают.

Изображение выглядит как текст, кроссворд, число, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

|  |
| --- |
| #include <iostream> #include <cstring>  using namespace std;  int main() {  setlocale(LC\_CTYPE, "rus");  char str[] = "abababbbabbbb"; // строка  char substr[] = "ab"; // подстрока, которую мы ищем  // функция strlen из библиотеки cstring получает длину строки.  int len\_str = strlen(str); // длина строки  int len\_substr = strlen(substr); // длина подстроки  int count = 0; // счетчик количества вхождений  for (int i = 0; i <= len\_str - len\_substr; i++) { // пока есть возможность поиска  bool found = true;  for (int j = 0; j < len\_substr; j++) { // цикл проверяет, соответствует ли текущая подстрока искомой подстроке  if (str[i + j] != substr[j]) {  found = false;  break;  }  }  if (found) {  count++;  cout << i << " ";  }  }  if (count == 0) {  cout << "Вхождений не найдено" << endl;  }  else {  cout << "\nОбщее количество вхождений: " << count << endl;  }  // вывод:  //0 2 4 8  //Общее количесвто вхождений: 4  return 0; } |

# Вектор

В стандартной библиотеке C++ *вектором* называется динамический массив, обеспечивающий быстрое добавление новых элементов в конец и меняющий свой размер при необходимости.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Сначала для создания вектора нам понадобится подключить библиотеку : <vector>.  
 Чтобы объявить вектор, нужно пользоваться конструкцией ниже:

|  |
| --- |
| vector < тип данных > <имя вектора>; |

Элементы вектора должны быть одинакового типа, и этот тип должен быть известен при компиляции программы. Он задаётся в угловых скобках после std::vector: например, std::vector<int> — это вектор целых чисел типа int, а std::vector<std::string> — вектор строк.  
 Мы можем инициализировать вектор одним из следующих способов:

|  |
| --- |
| std::vector<int> v1; // пустой вектор std::vector<int> v2(v1); // вектор v2 - копия вектора v1 std::vector<int> v3 = v1; // вектор v3 - копия вектора v1 std::vector<int> v4(5); // вектор v4 состоит из 5 чисел, каждое число равно 0 std::vector<int> v5(5, 2); // вектор v5 состоит из 5 чисел, каждое число равно 2 std::vector<int> v6{ 1, 2, 4, 5 }; // вектор v6 состоит из чисел 1, 2, 4, 5 std::vector<int> v7 = { 1, 2, 3, 5 }; // вектор v7 состоит из чисел 1, 2, 4, 5 |

Для обращения к элементам вектора можно использовать разные способы:

* **[index]**: получение элемента по индексу (также как и в массивах), индексация начинается с нуля
* **at(index)**: функция возвращает элемент по индексу
* **front()**: возвращает первый элемент
* **back():** возвращает последний элемент

# Задания:

**Задание 1.**

В вашей бд сделайте так, чтобы строковые поля в структуре могли состоять из нескольких строк. Перепишите ввод этих элементов используя getline.

Например: ФИО клиента записывается не как три отдельные поля в структуре: Фамилия, Имя, Отчество, а как одно и вводится с клавиатуры сразу.

Код:

#include <iostream>

#include <string>

#include <time.h>

#include <windows.h>

using namespace std;

struct person {

string surname;

string phone;

string address;

int out\_money;

bool diskountt = false;

};

struct person clients[100];

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

const int diskount = 200;

int n = 0;

int choice = 1;

while (choice != 0) {

cout << "1 - ввод элементов структуры с клавиатуры\n";

cout << "2 - вывод элементов структуры в консольное окно\n";

cout << "0 - Выход из программы\n\n";

cout << "Введите номер операции: ";

cin >> choice;

cin.ignore();

if (choice == 1) {

cout << "Введите данные: Фамилию, Имя,Контактный телефон, адрес" << endl;

cout << "Введите вашу фамилию и имя: ";

getline(cin, clients[n].surname);

cout << "Введите ваш контактный телефон: ";

cin >> clients[n].phone;

cin.ignore();

cout << "Введите ваш адрес: ";

getline(cin, clients[n].address);

clients[n].out\_money = rand() % 1000;

if (clients[n].out\_money >= diskount) {

clients[n].diskountt = true;

cout << "Поздравляем, вы получаете скидку в 10%!" << endl;

}

n++;

}

if (choice == 2) {

cout << "База данных" << endl;

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

cout << clients[i].surname << " ";

cout << clients[i].phone << " ";

cout << clients[i].address << " ";

if (clients[i].diskountt) {

cout << "Имеется скидка в десять процентов.";

}

cout << endl;

}

}

}

return 0;

}

Пример работы кода:

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Задание 2.**

Для вашей бд осуществите фильтрацию по строчным элементам структуры (например поле имя клиента или название продукта и тд). Фильтрация должна работать следующим образом: при вводе с клавиатуры подстроки, в консоль выводятся элементы массива структур, содержащие данную подстроку в поле, по которому происходит фильтрация.

Код:

#include <iostream>

#include <string>

#include <time.h>

#include <windows.h>

using namespace std;

struct person {

string surname;

string phone;

string address;

int out\_money;

bool diskountt = false;

};

struct person clients[100];

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

const int diskount = 200;

int n = 0;

int choice = 1;

while (choice != 0) {

cout << "1 - ввод элементов структуры с клавиатуры\n";

cout << "2 - вывод элементов структуры в консольное окно\n";

cout << "3 - фильтрация данных по адресу\n";

cout << "0 - Выход из программы\n\n";

cout << "Введите номер операции: ";

cin >> choice;

cin.ignore();

if (choice == 1) {

cout << "Введите данные: Фамилию, Имя,Контактный телефон, адрес" << endl;

cout << "Введите вашу фамилию и имя: ";

getline(cin, clients[n].surname);

cout << "Введите ваш контактный телефон: ";

cin >> clients[n].phone;

cin.ignore();

cout << "Введите ваш адрес: ";

getline(cin, clients[n].address);

clients[n].out\_money = rand() % 1000;

if (clients[n].out\_money >= diskount) {

clients[n].diskountt = true;

cout << "Поздравляем, вы получаете скидку в 10%!" << endl;

}

n++;

}

if (choice == 2) {

cout << "База данных" << endl;

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

cout << clients[i].surname << " ";

cout << clients[i].phone << " ";

cout << clients[i].address << " ";

if (clients[i].diskountt) {

cout << "Имеется скидка в десять процентов.";

}

cout << endl;

}

}

if (choice == 3) {

cout << "Введите адрес " << endl;

bool ed = false;

string s;

cin >> s;

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

if (clients[i].address == s) {

ed = true;

cout << clients[i].surname << " ";

cout << clients[i].phone << " ";

cout << clients[i].address << " ";

if (clients[i].diskountt) {

cout << "Имеется скидка в десять процентов.";

}

}

}

if(!ed) cout << "Отсутствуют люди с таким адресом";

}

}

return 0;

}

Пример работы кода:

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Задание 3.**

Написать программу, в которой нужно создать два вектора и объединить их, вывести на экран все уникальные элементы.

Код:

#include <iostream>

#include <vector>

#include <string>

using namespace std;

int main() {

setlocale(LC\_ALL, "Ru");

vector<int> arr;

vector<int> arr1;

vector<int> arr2;

int n, n1;

cout << "Введите количество элементов первого массива: ";

cin >> n;

cout << "Введите количество элементов второго массива: ";

cin >> n1;

cout << "Введите первый массив: ";

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

int q;

cin >> q;

arr.push\_back(q);

}

cout << "Введите второй массив: ";

for (int i = 0; i < n1; i++) {

int q;

cin >> q;

arr1.push\_back(q);

}

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

arr2.push\_back(arr[i]);

}

for (int i = 0; i < n1; i++) {

bool flag = false;

for (int j = 0; j < n; j++) {

if (arr1[i] == arr[j]) {

flag = true;

break;

}

}

if (!flag) {

arr2.push\_back(arr1[i]);

}

}

cout << "Неповторяющиеся элементы: ";

for (int i = 0; i < arr2.size(); i++) {

cout << arr2[i] << " ";

}

cout << endl;

return 0;

}

Пример работы кода: Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

**Задание 4.**

Создать матрицу из массивов vector. В данной матрице удалить любую одну строку с наибольшим количеством нулей.

#include <iostream>

#include <vector>

using namespace std;

int main() {

setlocale(LC\_ALL, "ru");

int rows, cols;

cout << "Введите количество строк: ";

cin >> rows;

cout << "Введите количество столбцов: ";

cin >> cols;

vector<vector<int>> matrix(rows, vector<int>(cols));

cout << "Введите элементы матрицы:" << endl;

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

cin >> matrix[i][j];

}

}

int max\_zeros = -1;

int row\_to\_delete = -1;

for (int i = 0; i < rows; i++) {

int zero\_count = 0;

for (int j = 0; j < cols; j++) {

if (matrix[i][j] == 0) {

zero\_count++;

}

}

if (zero\_count > max\_zeros) {

max\_zeros = zero\_count;

row\_to\_delete = i;

}

}

if (row\_to\_delete != -1) {

for (int i = row\_to\_delete; i < rows - 1; i++) {

matrix[i] = matrix[i + 1];

}

matrix.pop\_back();

}

cout << "Матрица после удаления строки с наибольшим количеством нулей:" << endl;

for (int i = 0; i < rows - 1; i++) {

for (int j = 0; j < cols; j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

return 0;

}

Пример работы кода:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.