МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Балтийский федеральный университет имени Иммануила Канта»

ОНК «Институт высоких технологий»

ОТЧЁТ О ПРОХОЖДЕНИИ

УЧЕБНО-ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКИ

на базе Высшей школы компьютерных наук и прикладной математики

Выполнил Большаков Игорь Романович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент очной формы обучения 1 курса

направления подготовки 01.03.02 Прикладная математика и информатика

профиль обучения «Анализ данных и машинное обучение»

Руководитель практики

Ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тарачков М.В.

г. Калининград 2022 г.

**СОДЕРЖАНИЕ**

Введение

Глава 1. Базовые понятия языка C++4

1. Базовые типы данных4
2. Условные операторы и циклы5
3. Массивы указатели и функции6

Глава 2. Задания на практику7

1. Задача 17
2. Задача 27
3. Задача 37
4. Задача 48
5. Задача 59
6. Задача 69

Глава 3. Выполнение заданий на практику11

Заключение

Список литературы

Приложения

**Введение**

Вид практики – Учебная технологическая (проектно-технологическая) практика (далее Учебная практика).

Цель учебной практики: получение первичных профессиональных умений навыков.

Задачи учебной практики:

* Закрепление и углубление теоретических знаний по программированию;
* Изучение основ объектно-ориентированного программирования в C++;
* Работа с памятью;
* Работа с библиотеками;
* Приобретение и развитие первичных профессиональных навыков и умений по прикладной математике и информатике.

**Глава 1. Базовые понятия Языка с++**

**Базовые типы данных.**

**Тип переменной** – характеристика, определяющая формат представления данных в памяти компьютера, множество допустимых значений этих данных и совокупность операций над ними. Зная тип переменной, компилятор выделит для нее необходимое количество ячеек памяти и будет знать, какого рода данные будут храниться в этой переменной.

|  |  |  |
| --- | --- | --- |
| Тип | Размер  (в байтах) | Диапазон |
| unsigned char | 1 | 0 – 255 |
| char | 1 | -128 – 127 |
| unsigned short | 2 | 0 – 65535 |
| short | 2 | -32 768 – 32 767 |
| unsigned int (16 разрядов) | 2 | 0 – 65 535 |
| unsigned int (32 разряда) | 4 | 0 – 4 294 967 295 |
| int (16 разрядов) | 2 | -32 768 – 32 767 |
| int (32 разряда) | 4 | -2 147 483 648 – 2 147 483 647 |
| unsigned long | 4 | 0 – 4294967295 |
| long | 4 | -2 147 483 648 – 2 147 483 647 |
| unsigned long long (С99) | 8 | 0 – 18 446 744 073 709 551 615 |
| long long (С99) | 8 | -9 223 372 036 854 775 808 –  9 223 372 036 854 775 807 |
| float | 4 | 1.2e-38 – 3.4e+38 |
| double | 8 | 2.2e-308 – 1.7e+308 |
| long double | 10 | 1.7e-4932 – 1.7e+4932 |

**Условные операторы и циклы**

**Условный оператор** позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действие. Структура условного оператора имеет следующий вид:

If(выражение) оператор; else оператор;

**Оператор for** формально можно записать следующим образом: for(выражение\_1; выражение\_2; выражение\_3) тело цикла

**Оператор while** имеет форму

While(условие) оператор-или блок;

**Оператор do-while** имеет форму:

Do оператор-или блок; while(условие);

Выражение в скобках может принимать либо нулевое (ложное), либо ненулевое (истинное) значение. Тело цикла будет выполняться, только если выражение истинно. Если же выражение ложно, то цикл завершается. Например, вывод на экран чисел 1, 2, …, 10 можно осуществить следующим образом:

Тело цикла составляет либо одна инструкция, либо набор инструкций, заключенных в фигурные скобки. Выражение\_1 обычно используется для инициализации начальных значений переменных, которые управляют циклом (управляющие переменные). Выражение\_2 определяет условия, при которых тело цикла будет выполняться. Выражение\_3 определяет изменение значений управляющих переменных. Цикл for выполняется по следующей схеме:

1. Вычисляется выражение\_1.

2. Вычисляется выражение\_2.

3. Если значение выражения\_2 является истинным (ненулевым), тогда выполняется тело цикла, а затем вычисляется выражение\_3 и осуществляется переход к выражению\_2. Если же значение выражения\_2 является ложным (нулевым), тогда выполнение оператора for завершается. При отсутствии выражения\_2 оно подразумевается всегда истинным.

Оператор break позволяет выйти из операторов цикла for, while, do-while и переключателя switch. Данный оператор дает возможность немедленного выхода из самого внутреннего цикла или переключателя.

Оператор continue вызывает преждевременное завершение выполнения тела цикла с переходом к следующему шагу итерации. Для циклов while и do-while это означает немедленный переход к проверке условия, а для цикла for – переход к выражению\_3. Оператор continue действует на самый внутренний цикл.

**Массивы и указатели. Функции.**

В отличие от обычных переменных, которые хранят одно значение, массивы используются для хранения целого набора однотипных значений. Массив – набор однотипных элементов, последовательно располагающихся в памяти компьютера. Для определения массива необходимо указать тип элементов, которые в него входят, и макси- мальное количество элементов, которое может быть помещено в массив.

Указатели представляют собой переменные, значениями которых являются адреса памяти. Если в "обычной" переменной содержится некоторое значение, то указатель содержит адрес той или иной переменной. "Обычная" переменная *непосредственно* ссылается на значение, указатель же ссылается на значение *косвенно*.

Синтаксис создания указателя имеет следующий вид: тип \*имя\_переменной,

где тип – тип переменной, адрес которой будет содержаться в указателе, имя\_переменной – имя переменной типа указатель, символ "звездочка" означает, что объявляемая переменная является указателем.

Функция – подпрограмма, которая организовывает тот или иной алгоритмический процесс и возвращает некоторое значение. Про- грамма на языке Си имеет по крайней мере одну функцию main(), которая автоматически вызывается при запуске программы. Сама функция main() может вызывать другие функции, а те, в свою очередь, са- ми могут вызывать те или иные функции.

Функция состоит из объявлений и операторов и предназначена для решения определенной задачи. Каждая функция должна иметь имя, которое используется для ее объявления, определения и вызова.

Когда имя функции встречается в программе, то управление передается к телу данной функции, то есть осуществляется вызов функции. При вызове функции ей при помощи аргументов (формальных параметров) передаются некоторые значения (фактические параметры), которые используются функцией во время ее работы. Любой аргумент функции может быть константой, переменной или выражением.

Тип возвращаемого значения объявляет тип значения, возвращаемого функцией. Если же функция не возвращает значения, тогда в качестве типа возвращаемого значения выступает пустой тип – void. Если функция не получает аргументы, список параметров также объявляется как void.

Функции могут возвращать значения. После обращения к функции она выполняет некоторые действия и в качестве результата своей работы может возвратить некоторое значение. С помощью инструкции return происходит возврат результата от вызываемой функции к вызывающей. После слова return может следовать любое выражение:

return выражение;

**Глава 2. Задания на практику**

**Задача №1**

**«RuCode»**

Принять участие в фестивале «RuCode» по направлению ИИ или алгоритмическое программирование. Это полезный навык, там будет обучение по использованию языков программирования для решения задач на алгоритмы.

Эта задача большая, т.е. она будет засчитана за несколько более маленьких задач (у Тарачкова М.В., если у вас другой преподаватель, уточните у него).

В случае вашего успешного выступления можно вести разговор о зачете всей учебной практики (у Тарачкова М.В., если у вас другой преподаватель, уточните у него).

**Задача №2**

**«Длинная арифметика»**

Даны 2 числа A и B, количество цифр в которых может достигать 1000. Необходимо реализовать операцию сложения и умножения над этими числами.

Будет засчитана только честная реализация (без спецбиблиотек и т.п.).

**Задача №3**

**«Обработка текста»**

Вариант 1

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие не менее четырех из пяти наиболее часто встречающихся букв текста, записать их заглавными буквами и указать после каждого такого слова в скобках найденные буквы. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

Вариант 2

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие подряд какие-нибудь две из четырех наиболее часто встречающихся букв текста, записать их заглавными буквами и указать после каждого такого слова в скобках найденные буквы. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

Вариант 3

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие ровно одну из 10 наиболее часто встречающихся букв текста записать их заглавными буквами и указать после каждого такого слова в скобках найденную букву. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

Вариант 4

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие две одинаковые буквы подряд, записать их заглавными буквами и указать после каждого такого слова в скобках найденные буквы. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

Вариант 5

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие не менее пяти из восьми наиболее часто встречающихся букв текста, записать их заглавными буквами и указать после каждого такого слова в скобках найденные буквы. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

Вариант 6

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие последовательность из четырех алфавитно упорядоченных букв, записать их заглавными буквами и указать после каждого такого слова в скобках найденные буквы. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

Вариант 7

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие не менее четырех различных гласных букв, записать их заглавными буквами и указать после каждого такого слова в скобках найденные буквы. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

Вариант 8

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие букву, не входящую ни в одно из слов текста с максимальной длиной, записать их заглавными буквами и указать после каждого такого слова в скобках найденные буквы. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

Вариант 9

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие более семи различных букв, выделить их и указать после каждого такого слова в скобках найденные буквы. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

Вариант 10

В текстовом файле input.txt записан русский текст. Найти в тексте слова, содержащие две одинаковые гласные буквы подряд, записать их заглавными буквами и указать после каждого такого слова в скобках найденные буквы. Полученный текст записать в файл output.txt. Весь текст, кроме найденных слов, должен остаться неизменным, включая и знаки препинания.

**Задача №4**

**«Битовая упаковка»**

Как известно, минимальный размер памяти, который мы можем прочитать, составляет 1 байт. Но иногда встречаются задачи, когда данные не такие большие, и даже 1 байта для них может быть много. Придумайте механизм упаковки нескольких значений в 1 байт.

Такой подход позволяет уменьшить размер используемой памяти и может быть полезен, например, при передаче данных по радиосвязи.

Порядок выполнения:

1. Определите минимальное количество бит на число. Если получается так, что в 1 байт вы можете записать 2.5 числа, например, то задействуйте 2 байта.
2. Разработайте операции добавления и извлечения чисел.

Вариант 1.

Целые числа от 0 до 10.

Вариант 2.

Целые числа от 0 до 63.

Вариант 3.

Целые числа от 0 до 34.

Вариант 4.

Целые числа от 0 до 3.

Вариант 5.

Целые числа от 0 до 17.

Вариант 6.

Целые числа от 0 до 2.

Вариант 7.

Целые числа от 0 до 31.

Вариант 8.

Целые числа от 0 до 15.

Вариант 9.

Целые числа от 0 до 5.

Вариант 10.

Целые числа от 0 до 8.

**Задача №5**

**«Умный динамический массив»**

Вам предлагается создать класс, который опишет умный динамический массив из целых чисел. Количество чисел заранее неизвестно. Ограничением служит размер доступной оперативной памяти компьютера, но использовать память все же надо рационально.

Что он должен уметь:

1. Добавлять элемент в конец.
2. Выдавать текущее количество элементов.
3. Выдавать элемент по индексу.
4. Изменять элемент по индексу.
5. Удалять элемент по индексу.
6. Вставлять элемент по индексу.

**Задача №6**

**«Операции над вектором»**

Изучите основные операции над вектором и продемонстрируйте их работу:

1. push\_back, erase, insert
2. emplace\_back
3. resize, reserve, shrink\_to\_fit, capacity
4. clear
5. оператор присваивания и []

Узнайте размер вектора и будьте готовы объяснить, почему именно так.

**Глава 3. Выполнение заданий на практику**

**Задача №1**

Нет решения

**Задача №2**

Основная идея заключалась в записывании длинных чисел в виде строки и в последующем записывании их в массив, что позволяло выполнять заданные операции над ними.

**Задача №3**

Считав текст из файла input.txt, я распределил знаки препинания, слова и четыре самые распространенные буквы в тексте в два динамических массива и строку соответственно, далее, с помощью циклов нашел слова, которые удовлетворяют условию задачи и записал их заглавными буквами, а также рядом с этими словами в скобках записал найденные буквы. После, я собрал все слова в общий текст и записал его в файл output.txt, сохранив при этом форматирование текста.

**Задача №4**

Я переводил десятичные числа в двоичные и затем с помощью операций работы с битами вписывал их в переменную типа данный unsigned short. Чтобы вывести эти числа обратно, я переводил их уже из двоичной системы в десятичную.

**Задача №5**

Написал класс массива, в котором описал заданные функции и продемонстрировал его работу

**Задача №6**

Я изучил и продемонстрировал работу указанных методов vector. А также выяснил чему равен sizeof (vector).

**Заключение**

В ходе практики были изучены основы языка программирования С++. Задачи были направлены на закрепление теоретического материала по типам данных, арифметическим, логическим операторам. Были изучены условия и циклы. Особое внимание уделялось работе с текстом, в частности массиву char.

Предлагалось решить задачи, связанные со спортивным программированием, длинной арифметикой, битовыми операциями.

В результате практики были усовершенствованы мои компетенции, закреплены теоретические навыки.

Во ходе практики я изучил такие разделы языка с++ как:

1. Типы данных (int, char, double, float, short и т.д.)
2. Циклы while, do while, for.
3. Условные операторы if else и switch.
4. Функции и процедуры.
5. Массивы.
6. Указатели и ссылки.
7. Основы объектно-ориентированного программирования.
8. Графическую библиотеку SFML.

И применил их в решение практических задач и выполнении итогового проекта.

Таким образом повысил свои навыки по программированию.

В течение практики все задачи были выполнены, а цели достигнуты.

**Список литературы**

**Перечень учебной литературы ресурсов сети «Интернет», необходимой для проведения практики**

1. Варфоломеева, Т. Н. Структуры данных и основные алгоритмы их обработки : учебное пособие / Т. Н. Варфоломеева. - Москва : ФЛИНТА, 2017. - 159 с. - ISBN 978-5-9765-3691-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/1860018 (дата обращения: 11.01.2022). – Режим доступа: по подписке.
2. Гданский, Н. И. Основы теории и алгоритмы на графах : учебное пособие / Н. И. Гданский. — Москва : ИНФРА-М, 2020. — 206 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-014386-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/978686 (дата обращения: 11.01.2022). – Режим доступа: по подписке.
3. Затонский, А. В. Программирование и основы алгоритмизации. Теоретические основы и примеры реализации численных методов: учебное пособие / А.В. Затонский, Н.В. Бильфельд. — 2-е изд. — Москва: РИОР : ИНФРА-М, 2022. — 167 с. — (Высшее образование). — DOI: https: //www.dx.doi.org/10.12737/20468. - ISBN 978-5-369-01195-9. - Текст: электронный. - URL: https://znanium.com/catalog/product/1860435 (дата обращения: 16.02.2022). – Режим доступа: по подписке.
4. Воронцова, Е. А. Программирование на С++ с погружением: практические задания и примеры кода - Москва: НИЦ ИНФРА-М, 2016. - 80 с. ISBN 978-5-16-105159-7. - Текст: электронный. - URL: https://znanium.com/catalog/product/563294 (дата обращения: 16.02.2022). – Режим доступа: по подписке.

**Перечень ресурсов информационно-телекоммуникационной сети «Интернет»**

1. Колдаев, В. Д. Структуры и алгоритмы обработки данных [Электронный ресурс]: учеб. пособие для вузов/ В. Д. Колдаев. - Москва: РИОР; Москва: ИНФРА-М, 2014. - 1 эл. опт. диск (CD-ROM), 294 с.: ил., табл. - (Высшее образование - бакалавриат). - Библиогр.: с. 285. - Лицензия до 23.06.2020 г. - ISBN 978-5-369-01264-2. - ISBN 978-5-16-009012-2: 15100.00 р. Имеются экземпляры в отделах /There are copies in departments: всего /all 2: ЭБС Кантиана(1), ч.з.N1(1) Свободны / free: ЭБС Кантиана(1), ч.з.N1(1)
2. Основы программирования. – режим доступа: http://www.intuit.ru/studies/courses/648/504/info
3. Видеолекции по курсу «Основы программирования». – режим доступа: http://www.youtube.com/watch?v=pxR3UoO9c9w
4. Сайт «Структуры и алгоритмы». – режим доступа: http://www.structur.h1.ru/
5. Структуры и алгоритмы обработки данных: Учебное пособие. – режим доступа: http://window.edu.ru/resource/820/44820

**Приложения**

**Приложение 1**

**Приложение 2**

int main() {

int arrayA[ARRAY\_SIZE] = { 0 };int arrayB[ARRAY\_SIZE] = { 0 };

int arrayResultSum[ARRAY\_SIZE] = {0 }; int arrayResultMultiply[ARRAY\_SIZE] = {0 };

std::ifstream file("(input.txt)");

std::string A; std::string B;

file >> A >> B;

file.close();

arrayA[0] = A.length();

arrayB[0] = B.length();

inputArray(A, arrayA);

inputArray(B, arrayB);

sumNumbers(arrayA, arrayB, arrayResultSum);

multiplyNumbers(arrayA, arrayB, arrayResultMultiply);

std::cout << "Sum of numbers:" << " ";

outputArray(arrayResultSum);

std::cout << "Multiplication of numbers:" << " ";

outputArray(arrayResultMultiply);

return 0;

}

**Приложение 3**

vector <string> arr;

vector <string> chars;

string simbols;

ifstream f("input.txt");

string temp = "";

string s;

string buf = "";

while (!f.eof())

{

s = "";

f >> s;

simbols.append(s);

simbol(s);

arr.push\_back(s);

}

char str[100];

int k = 0;

f.seekg(0);

while (!f.eof()) {

f.getline(str, 100);

for (int i = 0; i < strlen(str); i++) {

if (Simbol(str[i])) {

temp.push\_back(str[i]);

k++;

}

if (strlen(str) == i + 1) {

temp.push\_back('\n');

}

if ((!Simbol(str[i]) && k != 0) || strlen(str) ==i+1) {

chars.push\_back(temp);

k = 0;

temp = "";

}

}

}

enumeration(simbols, buf);

enumeration(simbols, buf);

enumeration(simbols, buf);

enumeration(simbols, buf);

k = 0;

string tmp = "";

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

{

for (int j = 0; j < arr[i].length(); j++)

{

if (buf.find(arr[i][j]) != -1)

{

k += 1;

tmp += arr[i][j];

}

else

{

k = 0;

tmp = "";

}

if (k == 2)

{

for (int g = 0; g < arr[i].length(); g++)

{

if (arr[i][g] >= 65 && arr[i][g] <= 90)

{

continue;

}

arr[i][g] -= 32;

}

arr[i] += "(" + tmp + ")";

k = 0;

tmp = "";

break;

}

}

k = 0;

}

ofstream file("output.txt");

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

file << arr[i] << chars[i];

}

file.close();

**Приложение 4**

std::vector<bool> ToBin(int num)

{

std::vector<bool> res;

res.resize(bit);

res.clear();

while (num > 0)

{

res.push\_back(num % 2);

num /= 2;

}

while (res.size() < bit)

res.push\_back(0);

return res;

}

int ToDec(std::vector<bool> n)

{

int res = 0;

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

{

res += n[i] \* pow(2, i);

}

return res;

}

void GiveNum(unsigned short& x, int index, int num)

{

std::vector<bool> tmp = ToBin(num);

std::vector<bool> nums;

nums.resize(bit);

nums.clear();

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

{

nums.push\_back(tmp[bit - 1 - i]);

}

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

{

if (nums[i])

{

x |= (1 << (bit \* index + i));

}

else

{

x &= ~(1 << (bit \* index + i));

}

}

}

int GetNum(unsigned short x, int index)

{

std::vector<bool> num;

num.resize(bit);

num.clear();

std::vector<bool> tmp;

tmp.resize(bit);

tmp.clear();

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

{

if (x & (1 << (bit \* index + i)))

{

tmp.push\_back(1);

}

else

{

tmp.push\_back(0);

}

}

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

{

num.push\_back(tmp[bit - i - 1]);

}

return ToDec(num);

**Приложение 5**

template<typename T>

class Array

{

private:

T\* m\_arr;

int m\_size{};

int m\_capacity{};

public:

Array()

{

m\_arr = new T[1];

m\_capacity = 1;

}

int Size() const

{

return m\_size;

}

int Capacity() const

{

return m\_capacity;

}

void AddMem()

{

m\_capacity \*= 2;

T\* tmp = m\_arr;

m\_arr = new T[m\_capacity];

for (int i = 0; i < m\_size; i++)

{

m\_arr[i] = tmp[i];

}

delete[] tmp;

}

void SetBack(const T& x)

{

if (m\_size >= m\_capacity)

{

AddMem();

}

m\_arr[m\_size++] = x;

}

void DeleteUnit(int a)

{

for (int i = a + 1; i < m\_size; i++)

{

m\_arr[i - 1] = m\_arr[i];

}

m\_size--;

}

T GetUnit(int i)

{

return m\_arr[i];

}

~Array()

{

delete[] m\_arr;

}

void ChangeUnit(int i, const T& x)

{

m\_arr[i] = x;

}

void AddByIndex(int j, const T& x)

{

if (m\_size + 1 >= m\_capacity)

{

AddMem();

}

T\* tmp = m\_arr;

for (int i = j; i < Size() + 1; i++)

{

m\_arr[i + 1] = tmp[i];

}

m\_arr[j] = x;

}

};

**Приложение 6**

std::vector <people> ListOfPeople;

people people1(17, 1, "Pasha");

people people2(28, 0, "Alexa");

people people3(40, 1, "Harry");

ListOfPeople.push\_back(people1);

ListOfPeople.push\_back(people2);

ListOfPeople.push\_back(people3);

std::cout << ListOfPeople[1].GetName() << "\n";

ListOfPeople.erase(ListOfPeople.begin()+1);

std::cout << ListOfPeople[1].GetName() << "\n";

people people4(10, 0, "Marlin");

ListOfPeople.insert(ListOfPeople.begin()+1, people4);

std::cout << ListOfPeople[1].GetName() << "\n";

ListOfPeople.emplace\_back(82, 1, "Oleg");

std::cout << ListOfPeople[3].GetName() << "\n";

std::cout << ListOfPeople.size() << "\n";

ListOfPeople.resize(3, people1);

std::cout << ListOfPeople.size() << "\n";

std::cout << ListOfPeople.capacity() << "\n";

ListOfPeople.reserve(5);

std::cout << ListOfPeople.capacity() << "\n";

ListOfPeople.shrink\_to\_fit();

std::cout << ListOfPeople.capacity() << "\n";

std::cout << sizeof(ListOfPeople) << "\n";

ListOfPeople[0] = people3;