**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

Кафедра информационных систем управления

**ОТЧЕТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

**Студент**

Царик Виталий Александрович

1 курс, 2 группа

**Преподаватель**

Гутников Сергей Евгеньевич

старший преподаватель кафедры

ИСУ

**Минск 2018**

Оглавление

[Оглавление 2](#_Toc516232353)

[Лабораторная работа №1 3](#_Toc516232354)

[Лабораторная работа №2 4](#_Toc516232355)

[Лабораторная работа №3 5](#_Toc516232356)

[Лабораторная работа №4 6](#_Toc516232357)

[Лабораторная работа №5 7](#_Toc516232358)

[Лабораторная работа №6 9](#_Toc516232359)

[Лабораторная работа №7 11](#_Toc516232360)

[Лабораторная работа №8 12](#_Toc516232361)

[Лабораторная работа №9 1](#_Toc516232362)4

[Лабораторная работа №10 1](#_Toc516232363)5

[Лабораторная работа №11 1](#_Toc516232364)7

[Реферат на тему Семантика перемещения. Перемещающий конструктор. Перемещающее присваивание 1](#_Toc516232365)9

Лабораторная работа №1

**Цель:**

Научиться создавать шаблоны функций для обработки массивов с элементами числового типа и массивов, элементы которых – строки типа char\*.

**Постановка задачи:**

Разработать шаблон функции, позволяющий найти в одномерном массиве максимальную монотонную подпоследовательность из подряд идущих элементов.

**Решение задачи:**

Сперва генерировался массив с заданными пользователем размерами, который заполнялся случайными значениями. После этого выводится получунная подпоследовательность, которая находятся с помощью шаблонной функции:

template <class Type>

void MaxMonotonicSubsequence(Type Sq[], int Size)

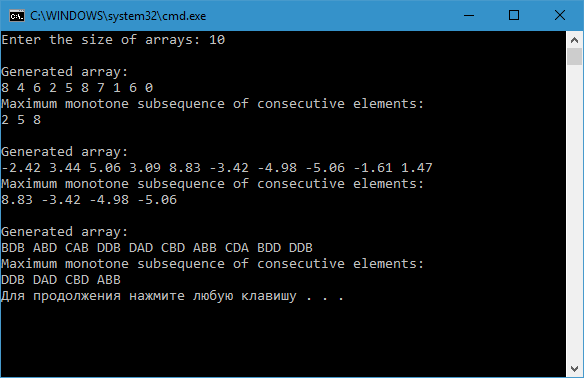


Рис.1 Результат работы программы

Лабораторная работа №2

**Цель:**

Научиться разрабатывать шаблоны и их спецификацию на примере реализации шаблона, аналогичного контейнеру STL.

**Постановка задачи:**

Разработать шаблон класса для работы со стеком, реалтзованным в динамическом массиве, и пользовательский тип – железнодорожный вагон (тип, цвет, вес, габариты).

**Решение задачи:**

В шаблоне класса Stack

template<typename Type> class Stack

определены методы pop, push, оператор присваивания, top, enpty:

void operator= (const Stack<Type>&);

void push(const Type&);

void pop();

Type top()const;

bool empty()const { return size == 0; }

В классе также присутствует спецификация шаблона данный методов для данных типа char\*.

Тестовая программа демонстрирует работу всех медов для данных типа int, char\* и пользовательского типа.

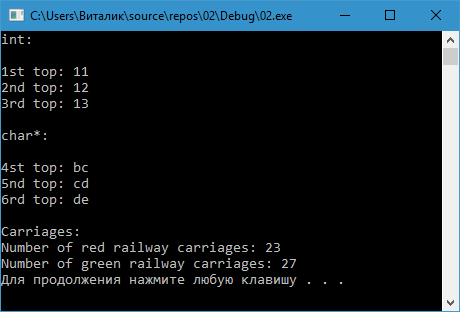


Рис. 2 Результат работы программы.

Лабораторная работа №3

**Цель:**

Ознакомиться с концепцией итератора и его реализацией

**Постановка задачи:**

Для шаблонного класса Stack разработать класс итератор, реализующий возможности, аналогичные итераторам STL.

**Реализация:**

Класс итератор содержит указатель Type\* ptr на элемент, с которым он работает и перегруженные операторы инкремента (постфиксный, префиксный), сравнения и разыменования. Оператор инкремента меняет адрес, на который указывает ptr, на адрес следующего элемента стека и возвращает полученное значение с помощью перегруженного оператора разыменования.

В классе Stack появились два новых метода begin и end, которые возвращают итераторы, указывающие на начало и конец очереди соответственно. Также добавлен метод print() для вывода стека на консоль

template<typename Type>

void Stack<Type>::print()

{

for (Stack<Type>::iterator it = begin(); it != end(); ++it)

cout << \*it << ' ';

}

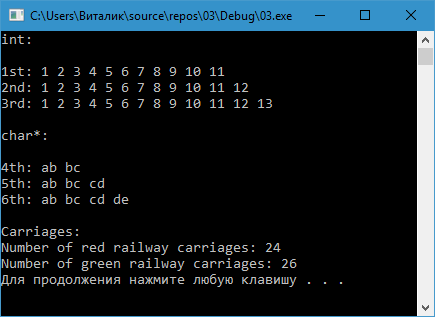


Рис. 3 Вывод стека с помощью итераторов.

Лабораторная работа №4

**Цель:**

Ознакомиться с паттерном Visitor и научиться его использовать.

**Постановка задачи:**

Для шаблонного класса стек:

* класс посетителя в соответствие с шаблоном Visitor
* методы доступа для посетителей (Accept(), …)
* определить методы min() и max() для нахождения минимального и максимального элементов шаблонного класса используя класс посетителя.
* определить методы imin() и imax() для нахождения минимального и максимального элементов шаблонного класса используя класс итератор
* создать тестовое приложение для тестирования разработанного кода

**Реализация:**

В класс стек был добавлен метод accept, имеющий вид

void accept(Visitor<Type> &v) { v.visit(this); }

и был создан шаблон класса Visitor, который создержит однин чисто виртуальный метод visit().

Реализованы методы min() и max(), которые находят минимальный и максимальный элемент класса стек используя класс Visitor:

min\_Visitor<int> min;

max\_Visitor<int> max;

S.accept(min);

S.accept(max);

А также, методы imax() и imin(), минимальный и максимальный элемент класса стек используя класс Iterator:

imin\_Visitor<int> imin;

imax\_Visitor<int> imax;

S.accept(imin);

S.accept(imax);

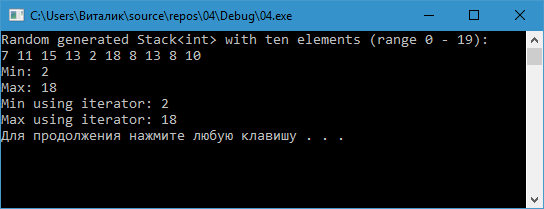


Рис. 4 Результат работы тестового приложения.

Лабораторная работа №5

**Цель:**

Научиться разрабатывать веб-страницы с использованием HTML и CSS.

**Постановка задачи:**

Разработать 4 HTML страницы, содержащую информацию о Вас: index.html, место рождения, место учебы, увлечения. Обеспечить переход между страницами с помощью ссылок.

**Реализация:**

В заголовке каждой страницы находится меню с ссылками на другие страницы:

<div id="Header"> <!-- графическое меню -->

<a href="#"> Main Page </a><span>|</span>

<a href="page1.html"> Place of Birth </a><span>|</span>

<a href="page2.html"> Education </a><span>|</span>

<a href="page3.html"> Hobby </a>

</div>

Основная информация на главной странице размещена в таблице. Также на стринеце присутствует текстовое поле для отправки письма на почту.

**Особенности реализации:**

Меню, которое двигается вместе со страницей получено с помощью использования следующих CSS правил для блока Header.

**#Header {**

**margin: auto;**

**font-family: monospace;**

**background-color: #E0F8F7;**

**width: 550px;**

**height: 40px;**

**text-align: center;**

**padding-top: 15px;**

**border: 1px solid black;**

**}**

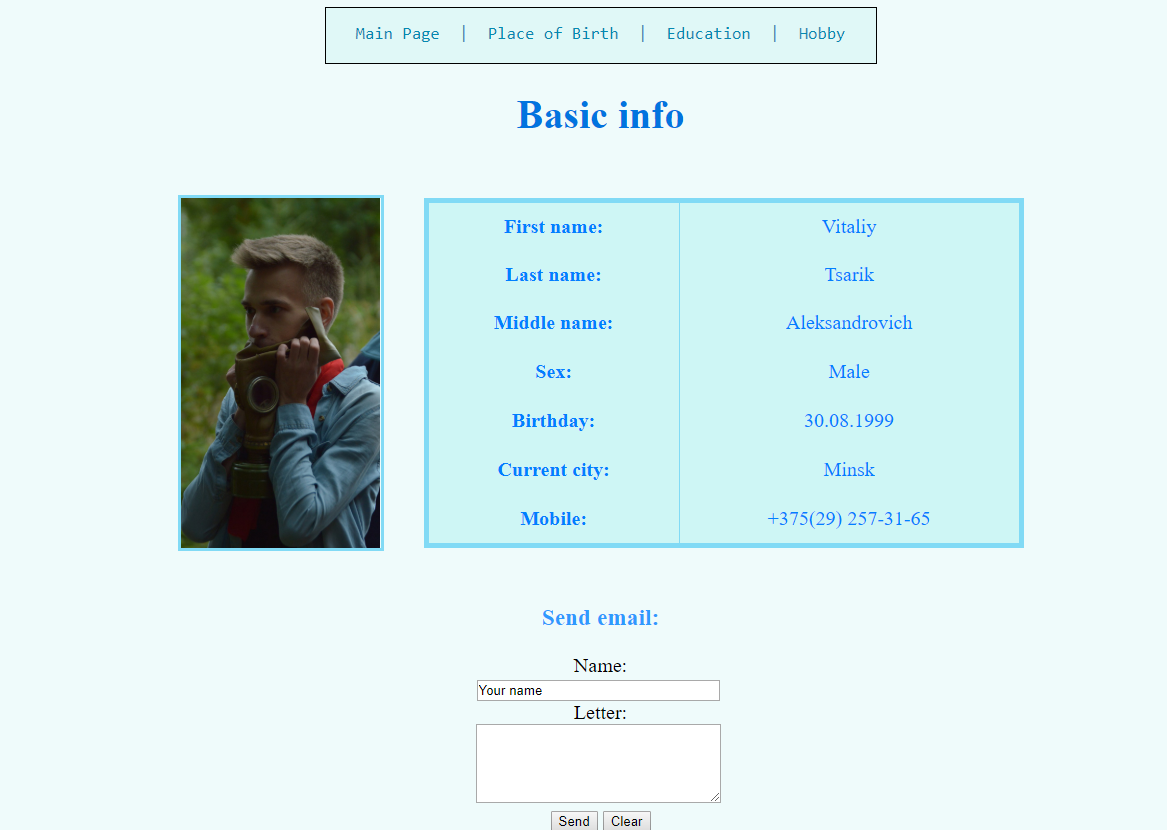


Рис. 5.1 Главная страница

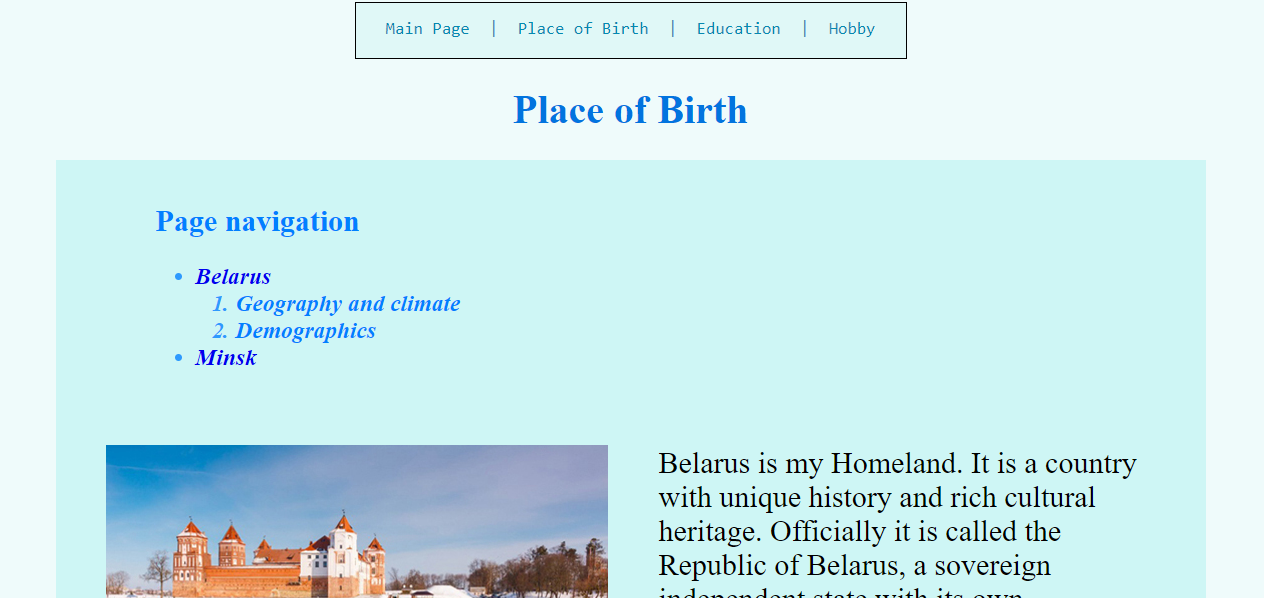


Рис. 5.2 Страница «Место рождения»

Лабораторная работа №6

**Цели:**

Научиться разрабатывать оконные приложения с использованием WinApi. Научиться работать с компонентами Button, RadioBox, ComboBox.

**Постановка задачи:**

Разработать приложение, соответствующее следующим требованиям:

1. Главное окно должно содержать меню.
2. Пункт меню Просмотреть открывает модальное диалоговое окно просмотра.
3. При выборе пункта меню Диаграмма главного окна в модальном диалоговом окне нарисовать диаграмму, отображающую соотношение каких-то числовых данных вашей задачи. Подписи диаграммы должны отображать всю информацию. Выбор цвета столбиков выполнить через пункт меню Цвет главного окна в модальном диалоге с использованием зависимых кнопок RadioButton из нескольких цветов и через стандартный диалог выбора цвета.
4. Выбор пункта меню Об Авторе главного окна отображает на экране немодальное диалоговое окно с вашими данными.
5. Использовать контекстное меню, дублирующее некоторые пункты главного меню. Например, пункты Диаграмма и Цвет. Контекстное меню отображается по клику правой кнопкой мыши в клиентской области главного окна.

**Реализация:**

Главное окно содержит меню с пунктами file, View, Diagrem, Color, About. По нажатию правой кнопки мыши, появляется контекстное меню с пунктами View, Diagram, Save, Close, Exit. В окне View (Рис. 6.1) данные представляются в ListBox и дополнительные данные выводятся в несколько EditBox-ов. Выбор шрифта реализован с помощью ComboBox. По нажатию на кнопку Add открывается диалоговое окно (Рис.6.2). Окно Diagram (Рис.6.3) отвечает за графическое представление данных. С помощью кнопки пункта меню color можно выбрать цвет диаграммы.

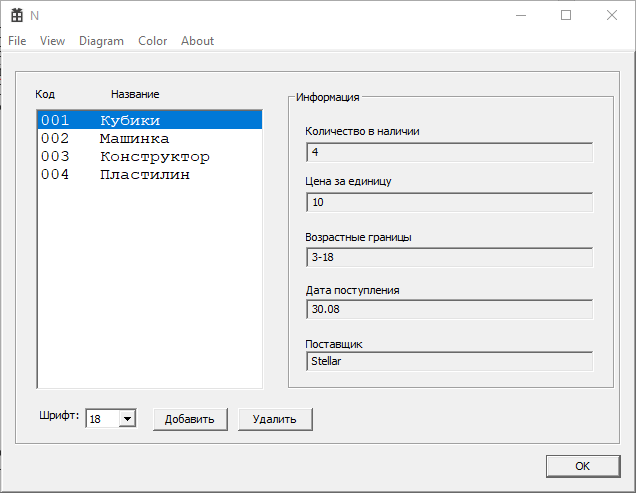


Рис. 6.1 Окно просмотра

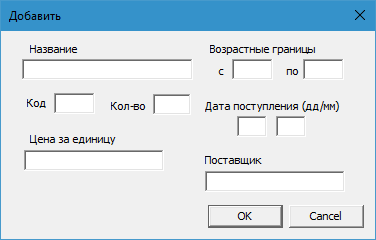


Рис. 6.2 Окно добавления элемента

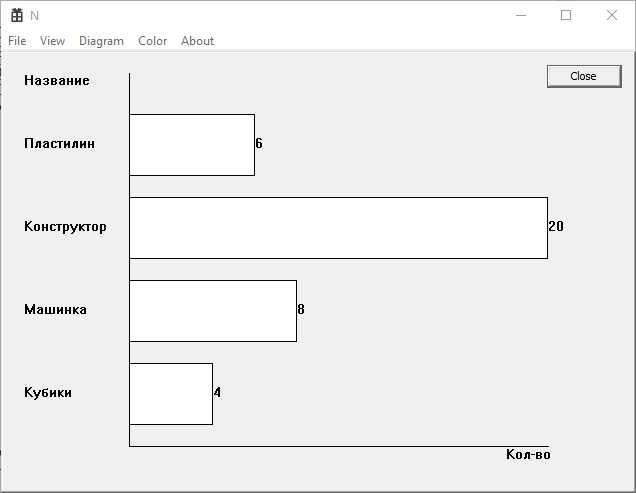


Рис. 6.3 Окно диаграммы

Лабораторная работа №7

**Цель:**

Ознакомиться с паттерном MVC.

**Постановка задачи:**

Переработать решение лабораторной работы №6 с использованием шаблона MVC.

**Реализация:**

Вся работа с данными реализуется в классе \_Data.

class \_Data // Model

Классы \_View и \_DiagramView отвечают за графическое отображение данных. Класс \_Controller реагирует на действия пользователя, а также отвечаеи за чтение из файла и запись в файл.

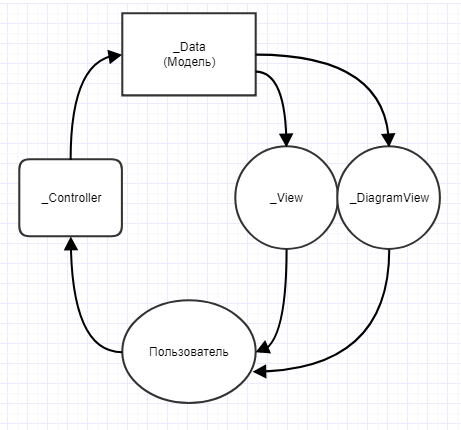


Рис. 7 Схема работы приложения

Лабораторная работа №8

**Цель:**

Ознакомиться с языком программирования C# и его базовыми понятиями, научиться работать с интерфейсами IComparer, IComparable и IDisposable.

**Постановка задачи:**

Создать интерфейс Body с методами вычисления площади поверхности и объёма. Создать класс Parallepiped (параллепипед) реализующий интерфейс Body.

Класс должен реализовать:

-интерфейс IComparer с возможностью выбора одного из полей для сравнения

-интерфейс IDisposable

-индексатор по всем полям

-метод для сохранения значений всех полей в строке текста (переопределить

ToString())

-конструктор или метод для инициализации объекта из строки текста

Создать консольное приложение, демонстрирующее использование класса. Создать небольшой массив объектов и напечатать отсортированными по выбранному полю.

**Реализация:**

Создан интерфейс Body c методами SurfaceArea(), Volume():

interface Body

{

float SurfaceArea();

float Volume();

}

Для задания объекта класс Parallepiped использовались три структуры Vector3 из System.Numerics.

private Vector3 OA { set; get; }

private Vector3 OB { get; set; }

private Vector3 OC { get; set; }

В классе Parallepiped определены следующие интерфейсы:

//IComparer

public int Compare(object x, object y)

//IComparable

public int CompareTo(object obj)

//IDisposable

public void Dispose()

Индексация по полям класса осуществляется с помощью следующей конструкции:

public string this[int idx]

Демонстрационная программа показывает работу функции CompareTo и демонстрирует сортировку массива из трёх объектов по различным полям.

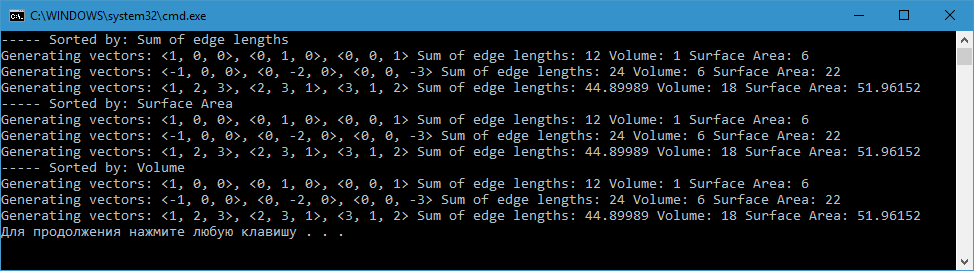


Рис. 8 Результат работы программы

Лабораторная работа №9

**Цель:**

Научиться использовать команды математического сопросцессора.

**Постановка задачи:**

Разработать консольное приложение. Функция представлена в виде своего ряда Тейлора. Используя встроенный ассемблер вычислить приближённое значение суммы этого бесконечного ряда. При вычислении следующего слагаемого использовать значение, полученное на предыдущей итерации.

**Реализация:**

Функция ASM\_Taylor принимает два параметра по значению: точность k и аргумент x.

Сначала расчитывается в зависимости от k и помещается в стек. Далее в стек заносится число x и счетчик i. И n-ое слагаемое считается по следующей формуле: . Затем происходит сравнение слагаемого, взятого по модулю, с точностью и переход в начало цикла, если слагаемое не меньше

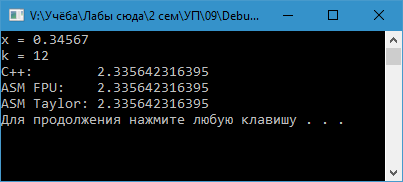


Рис. 9.1 Работа программы с положительным значением

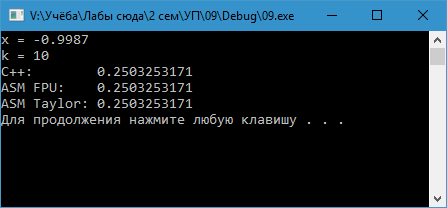


Рис. 9.2 Работа программы со значением близким к границе

Лабораторная работа №10

**Цель:**

Ознакомиться с базовыми возможностями библиотеки WindowsForms.

**Постановка задачи:**

Построить график зависимости числа итераций от точности вычисления суммы бесконечного ряда из задания 9. Перед построением графика данные необходимо записать в файл.

**Реализация:**

Программа из предыдущего задания была модифицирована. Она считает количество итераций для точности от 2 до 13 знаков и записывает в файл значение x и количество итераций для каждой точности.

При запуске программы, чертятся координатные оси и координаткная сетка, расмечается маштаб в зависимости от наибольшего количества итераций. После этого, с помощью внеэкранного буфера рисования и функции DrawLine(), программа строит график по заданным значениям.

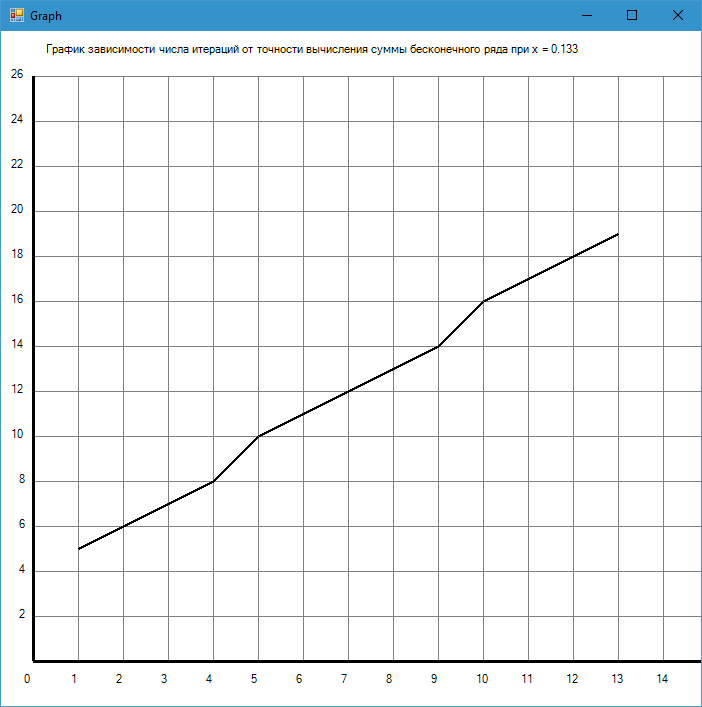


Рис. 10.1 График для x = 0.133

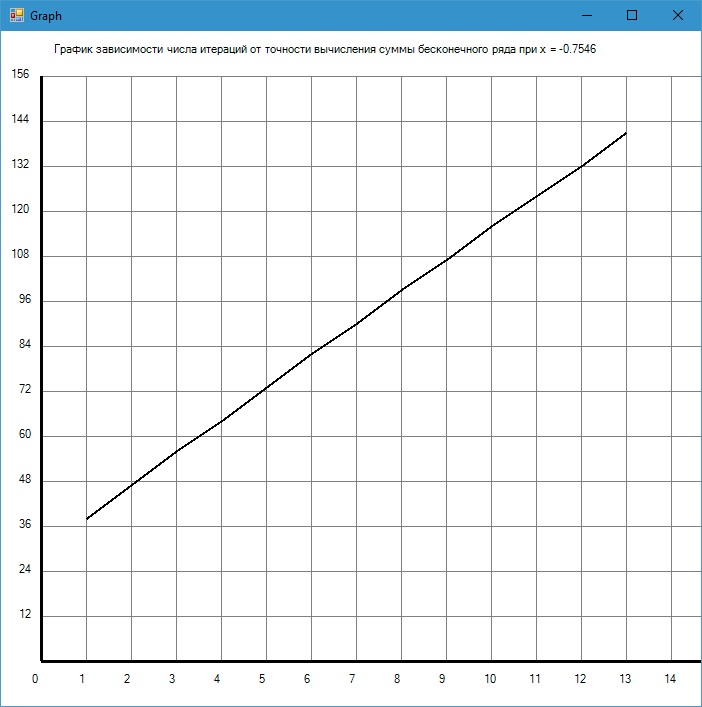


Рис. 10.2 График для x = -0.7546

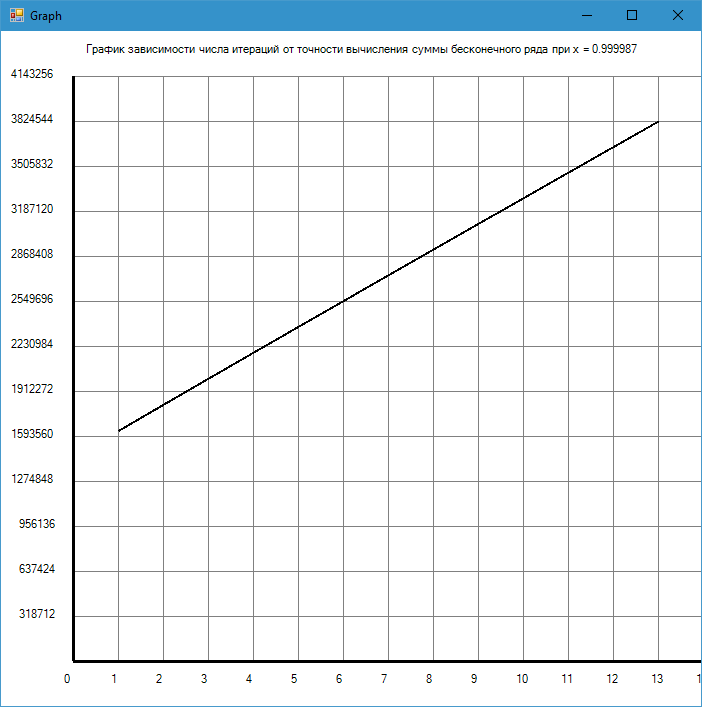


Рис. 10.3 График для x = 0.999987

Лабораторная работа №11

**Цель:**

Ознакомиться с базовыми возможностями библиотеки WindowsForms и методами сериализации и десериализации данных.

**Постановка задачи:**

1. Спроектировать классы, указанные в задаче. Организовать хранение объектов впамяти в виде динамического массива (ArrayList). Используя механизм сериализации организовать хранение объектов в файле.
2. Разработать оконное приложение WindowsForms для просмотра/редактирования записей файла. Используйте стандартные диалоги для выбора файла.
3. Предусмотреть обработку исключений и ошибок без аварийного завершения приложения.

**Реализация:**

Главное окно (Рис. 11.1) содержит меню (Рис. 11.2), Label-ы с перечислением имен элементов и элементы управления для взаимодействия с пользователем.

При открытии файла происходит десерилизация ArrayList и заполнение данных класса данными из файла.

В случае закрития без сохранения изменеий будет вызван предупреждающий диалог

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if (Data.SavedToFile == false && Data.lastIndex != -1)

if (MessageBox.Show("Exit with unsaved changes?", Text,

MessageBoxButtons.OKCancel, MessageBoxIcon.Question)

== DialogResult.Cancel)

{

e.Cancel = true;

}

}

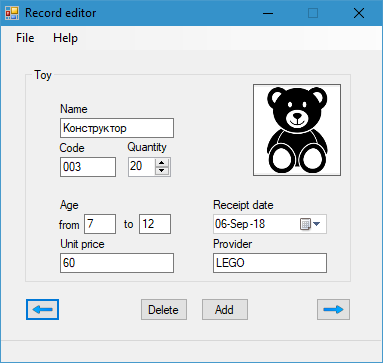


Рис. 11.1 Главное окно

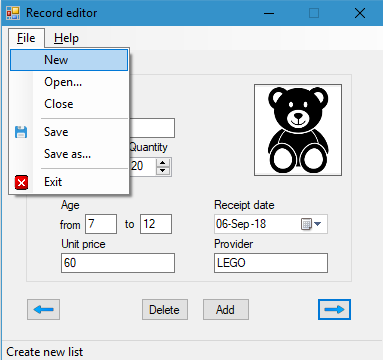


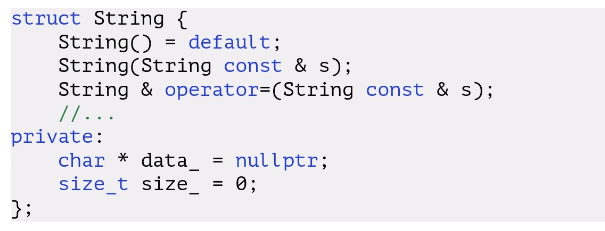
Рис. 11.2 Демонстрация меню

РЕФЕРАТ

Семантика перемещения. Перемещающий конструктор. Перемещающее присваивание

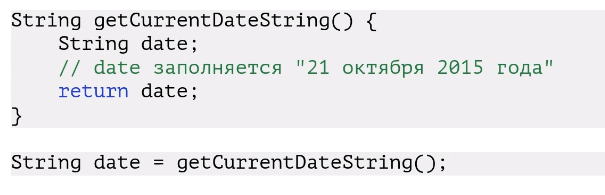
1. Проблема: излишнее копирование

Копирование объекта часто является дорогой операцией. В качестве демонстрации можно привести простой класс string (Прим.1.1). Чтобы скопировать объект этого типа необходимо выделить буфер нужного размера в новом объекте и посимвольно скопировать саму строку. Аналогично для оператора присваивания, только буфер сначала придётся очистить. Т.е. копирование – дорогая операция (пропорциональна длине строки).



Прим.1.1

Пусть теперь есть функция, которая возращает строчку с датой. (Прим. 1.2)



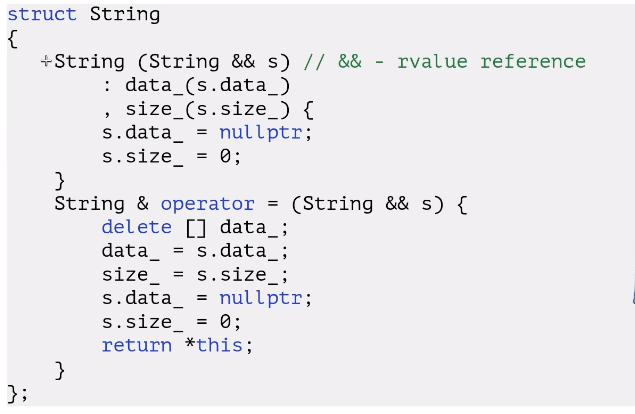
Прим. 1.2

При вызове функции, после заполнения переменная date будут скопирована в место зарезервированное для возращаемого значения. Далее, чтобы сохранить полученное значение, создадим (уже вне функции) переменную date и проинициализируем её значением возращаемым из функции. Т.е. ещё один вызов конструктора копирования. Итого, два ненужных копирования.

В стандарте C++11 есть специальный механизм, позволяющий избежать такого лишнего копирования.

1. Перемещающий конструктор и перемещающий оператор присваивания

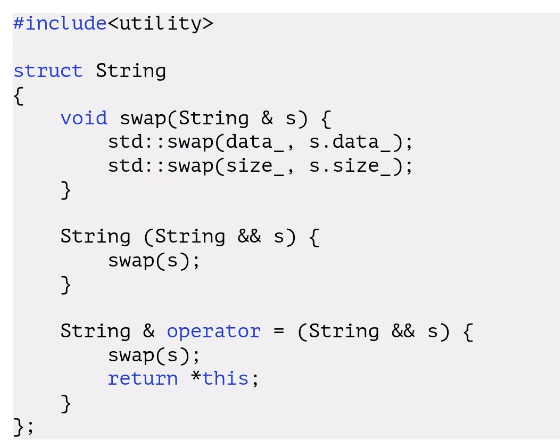
В стандарте появились два новых метода: перемещающий конструктор и перемещающий оператор присваивания. Их сигнатуры можно посмотреть на примере нашей строчки (прим. 2).



Прим. 2.1

В чём отличие от контруктора копирования? В контруктора копирования объект передаётся по константной ссылке, т.е. его изменять нельзя. В перемещающем конструкторе же – по неконстантной ссылке нового типа (rvalue reference), которая задаётся &&. Получается мы можем менять исходный объект, но его необходимо оставить в согласованном состаянии. Т.е. в перемещающем конструкторе не обязательно копировать данные, а можно просто “оотбрать” их у исходного объекта. В случае в нашей строчкой, это делатся легко: нужно скорировать указатель на буфер и скопировать размер буфера. А исходный объект остаётся в состоянии пустой строки. Выгода, по сравнению с коструктором копирования, очевидна – дорогое посимвольное копирования заменилось на копирования указателя. В перемещающем операторе присваивания ситуация аналогичная: опять же, сначала нужно удалить текущий буфер, затем отобрать данные.

Ниже приведёт более лаконичный пример реализации перемещающего конструктора и оператора присваивания с использованием метода swap. Причём эта реализации немного отличается в работе. В предыдущем примере после перемещения исходный объект остаётся в состоянии пустой строки. А данном случае объекты обмениваются данными, а исходная строка после перемещение может не быть пустой и будет висеть в какое-то время в памяти. Тут важно, чтобы при объявлении строки у неё было какое-то значение по умолчанию.

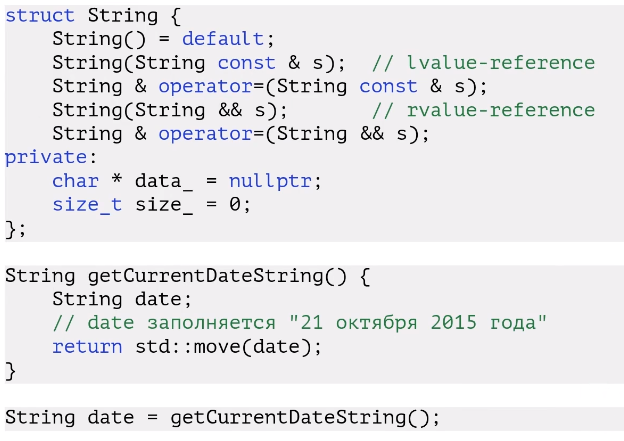


Прим.2.2

1. Использование перемещения

Дополним пример 1 перемещающим конструктором и перемещающим оператором присваивания (прим.3). Теперь вместо двух вызовов конструктора копирования будет вызываться перемещающий конструктор.

Итого, дорогое посимвольное копирование (линейная сложность) заменилось на дешёвое копирование указателя и размера буфера (константная сложность).



Прим. 3

Стоить заметить, что в данном примере не обязательно явно прописывать std::move(), т.к. в функции по значению возвращается локальный объект data.

Когда используется перемещающие методы:

* если передавать объект при помощи std::move()
* если передавать временный объект
* если из функции по значению возвращается локальный объект функции