СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Учебная практика была пройдена в Амурском Государственном Университете с «17» июня 2024 года по ««29» июня 2024 года. Юридический адрес Амурского Государственного Университета: Игнатьевское шоссе 21, город Благовещенск, Амурская область.

Руководителем практики являлась Никифорова Лариса Владимировна.

Целью учебной практики заключалась в овладении студентами профессиональной деятельности согласно требованиям к уровню подготовки бакалавров по направлению подготовки 09.03.04 Программная инженерия.

Задачами практики являются:

Углубление знаний по дисциплинам, полученным за время обучения на первом курсе, таких как: «Программирование», «Информатика», «Компьютерные и информационные технологии в профессиональной деятельности», «Цифровая грамотность», «Линейная алгебра и теория матриц».

Развитие практических навыков разработки прикладного программного обеспечения и применения современных инструментальных средств для их создания;

Развитие практических навыков инсталляции и настройки программного обеспечения общего назначения и специализированных программ;

Формирование навыков подготовки и систематизации необходимых материалов и научно-технической информации для выполнения задания; - создание условий для практического применения знаний в области общепрофессиональных, специализированных компьютерных и математических дисциплин.

Формирование и совершенствование базовых профессиональных навыков и умений в области применения современных информационных технологий.

Формирование информационной компетентности с целью успешной работы в профессиональной сфере деятельности.

Приобретение навыков создания отчетов, в том числе и научно-технических, обеспечение успеха дальнейшей профессиональной карьеры.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

При выполнении учебной практики было проведено ознакомление со следующими разделами её программы:

Со структурой и основными принципами организации работы во время учебной практики Амурского Государственного Университета.

С видами деятельности, связанными с направлением подготовки 09.03.04 Программная инженерия.

С интерфейсом программного обеспечения, необходимого для разработки алгоритма решения, кодирования программы и ее отладки.

С подходами и методологиями для успешного выполнения индивидуального задания.

В процессе учебной практики была получена информация, необходимая для успешного выполнения индивидуального задания. Проведена инсталляция программного обеспечения Microsoft Visual Studio, а также изучена теоретическая часть, необходимая для событийно-управляемого программирования в среде разработки Visual Studio.NET.

1.1 Особенности создания Windows-приложений на языке C++ в среде Visual Studio 2022

Среда разработки Visual Studio 2022 позволяет создавать, отлаживать и проводить тестирование приложений на Windows. Данная среда разработки поддерживается такими версиями Windows как Windows 11 и Windows 10. Одной из самых важных преимуществ этой среды является поддержка различных инструментов для управления зависимостями и сборкой проекта.

Visual Studio 2022 предоставляет мощные инструменты для разработки

Windows-приложений на C++. Для начала работы необходимо было сделать следующее:

Установить Visual Studio 2022 с компонентами для разработки на C++. Для выполнения задания было необходимо было установить компонент "Desktop development with C++". Данный компонент позволяет контролировать исходный код, управлять рабочими элементами, учитывать синтаксис при редактировании кода.

Настроить среду разработки, выбрав необходимые компоненты и SDK для Windows. Для возможности работы с Windows Form необходимо было установить расширение C++/CLI.

1.2 Интеграция C++ с Visual Studio.NET

Среда Visual Studio.NET содержит удобные средства разработки Windows-приложений, позволяющие избавить программиста от рутинной работы. Однако при интеграции среды .NET в язык программирования C++, возникают следующие особенности:

Среда разработки Visual Studio .NET поддерживает C++, но для работы необходимо использовать расширение C++/CLI. C++/CLI – это расширение позволяющее использовать возможности платформы .NET Framework для языка программирования C++ [1].

Необходимо знать, как использовать специальные инструменты и функции, так как среда разработки Visual Studio .NET имеет собственную функциональностью и инструменты.

У Visual Studio .NET имеется собственная система сборки и управления проектами, которая отличается от среды разработки C++. Именно из-за этого программисту необходимо знать специализированные функции и инструменты.

При применении расширения C++/CLI программист может использовать стандартный функционал C++ с возможностью использовать

управляемые типы и классы. Данная возможность позволяет создавать типы-обертки, которые можно будет потом использовать в любом языке NET.

Платформа .NET состоит из следующих компонентов:

Common Language Runtime (CLR) - реализацией спецификации CLI. Так как приложения .NET закодированы на языке CIL, то основная задача CLR заключается в обеспечении выполнения приложений .NET.

Библиотека классов Framework (FCL) - это компонент Microsoft .NET Framework, реализующий систему виртуального выполнения CLI. FCL необходим для реализации базовых стандартных библиотек CLI.

Что бы связать код на C++ и среду NET необходимо чтобы все основные части кода находились вне функции main(), из-за того что во время компиляции функция main будет заменена. Это можно достичь вынесением алгоритмов в отдельные функции и типы данных.

При использовании CLR необходимо помнить, что определенная часть синтаксиса подчиняется своим особым правилам. Например, при создании объекта управляемого класса в динамической памяти необходимо знать, что вместо «new» нужно использовать «gcnew». Приписка «gc» позволяет указать компилятору то, память выделяется под управляемый класс, который не требует ручной очистки и данный объект будет подчинятся правилам среды CLR [3].

Типы данных, реализованных в C++ при использовании в управляемых классах CLR необходимо конвертировать. Конвертация должна происходить в явном или не явном виде. Использование неявного вида не допустимо при возможности потери информации.

Среда выполнения Windows и среда CLR представляют свои типы данных в виде объектов, управление выделяемой памятью которых осуществляется автоматически. Это значит, что в случае выхода переменной за пределы области видимости или завершении работы приложения явно отменять память для переменной не потребуется.

2 РАЗРАБОТКА ПРОГРАММЫ И ИНТЕРФЕЙСА

2.1 Описание программы

Обозначение и наименование программы: «Буханов Денис Евгеньевич, группа 3105 об, 3 вариант».

Программное обеспечение, необходимое для работы программы: Операционная система Windows 10 или Windows 11, с поддержкой архитектуры x86-64

Язык программирования: C++/CLI.

Функциональное назначение: генерация случайных и заданных пользователем значений массива, сортировка массива и вывод его на экран.

Описание логической структуры: код включает в себя метод «get_inform» класса «InputForm» который генерирует массив и метод «sort» класса «My_arr» который необходим для сортировки массива [2].

Метод «get_inform» генерирует массив по следующему правилу:

Введенный параметр является центральным и последним элементом массива, остальные элементы создаются с помощью счетчика случайных чисел. Всего в массиве содержится 25 элементов.

Метод «sort» выполняет сортировку массива следующем образом: если положительных элементов массива больше чем отрицательных, то отсортированные по убыванию положи-тельные элементы расположить в начале массива. Иначе в начале массива расположить отсортированные отрицательные элементы.

Входные и выходные данные: Метод «get_inform» класса «InputForm» при своей работе считывает данные с формы которую заполнил пользователь, генерирует вектор размерностью 25 и возвращает данный вектор. Если флажок

типа CheckBox будет иметь значение «Истинна», то в вектор будут добавлены числа, созданные с помощью генератора случайных чисел. После полученный вектор используется при создании объекта класса «My_arr». Метод «sort» класса «My_arr» позволяет отсортировать вектор по выше изложенному правилу [4]. В таблице 1 перечислены все вышеуказанные элементы, их идентификаторы и типы.

Таблица 1 – Основные переменные программы

Идентификатор	Тип	Хранимые данные
min_range	int	Минимальный диапазон случайных чисел
max_range	int	Максимальный диапазон случайных чисел
items	vector<int>	Последовательный контейнер вектор целочисленных значений
my_arrey	My_arr<int>*	Массив содержащий последовательный контейнер

В последствие, для интеграции кода C++ с Средой Visual Studio.NET данные переменные будут переконвертированы в аналогичные типы данных, поддерживаемые языком C++/CLI. В таблице 2 указаны аналоги этих переменных в соответствующем порядке.

Таблица 2 – Переменные после конвертации

Идентификатор	Тип	Аналогичная переменная
min_range_random->Value	Int32	min_range
max_rande_random->Value	Int32	max_range

2.2 Тестирование программы

Согласно алгоритму программы, в результате её выполнения можно будет получить 2 массива с 25 значениями по указанным выше правилам. Предполагается, что при вводных данных, приведённых на рисунке 1, результат будет следующим: необработанный массив состоит из 25 элементов, центральный и последний элемент имеет значение «89», все остальные элементы задаются случайно в диапазоне от [-100; 100].

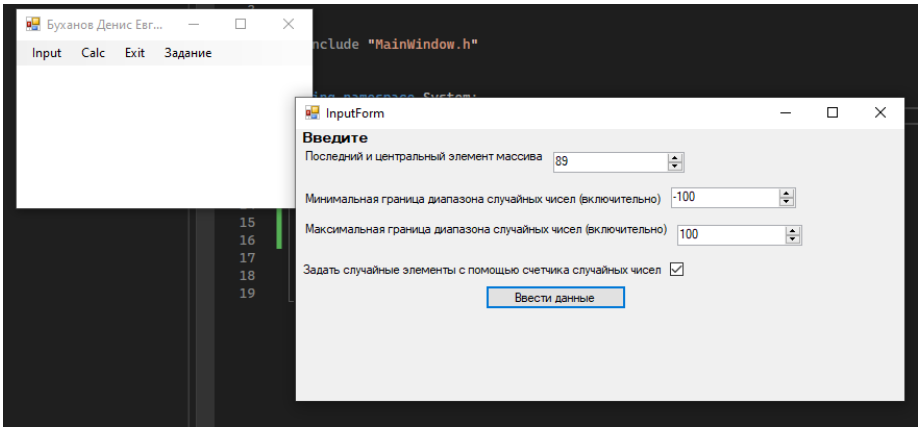

Рисунок 1 – Скриншот ввода данных в программу.

Результат обработанного массива будет являться отсортированный массив, начинающийся с отсортированных в порядке убывания отрицательных чисел если отрицательных чисел в массиве больше, чем положительных или с отсортированных в порядке убывания положительных чисел если положительных чисел в массиве больше, чем отрицательных. На рисунке 2 видно, что программа выдала ожидаемый результат.

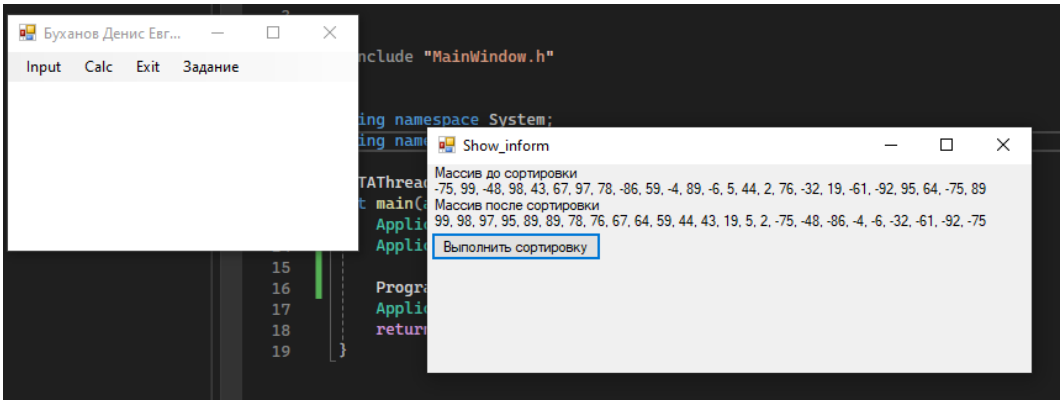

Рисунок 2 – Скриншот выполнения программы.

Полученный массив имеет 25 элементов, все случайные значения не выходят за приделы диапазона, отсортированный массив начинается с максимального числа «99».

2.3 Руководство пользователя (оператора)

Условия применения: программное обеспечение может эксплуатироваться и выполнять заданные функции при соблюдении требований предъявляемых к техническому, системному и прикладному программному обеспечению.

Запуск программы: для начала работы с программой необходимо открыть скомпилированный файл программы с расширением «exe». После открытия программы, программа отрисует интерфейс главного окна (рис. 3)

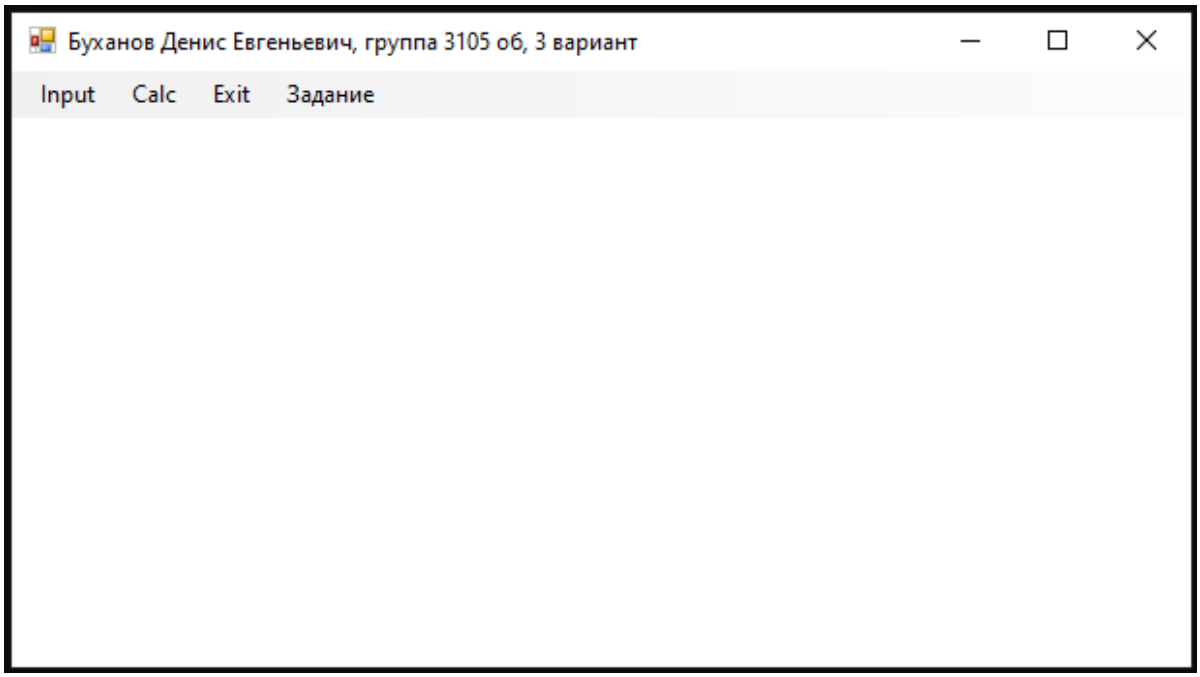

Рисунок 3 – интерфейс главного окна.

Интерфейс главного окна содержит панель меню с следующими пунктами:

Input – пункт меню, вызывающий форму для ввода данных в программу.

Calc – данный пункт меню необходим для вызова формы для вывода и сортировки массива.

Exit – пункт меню, завершающий приложение.

Задание – пункт меню, вызывающий форму для вывода задания. В данной форме описан вариант индивидуального задания.

Для выполнения программы необходимо проделать следующие действия:

Нажать пункт меню «Input» (рис. 4) и в открытом меню пользователю нужно ввести последний и центральный элемент массива, минимальную и максимальную границу диапазона случайных чисел и отметить флажок об использовании случайных чисел. После того как пользователь ввел информацию в виджеты, необходимо нажать на кнопку «Ввести данные» и закрыть диалоговое окно.

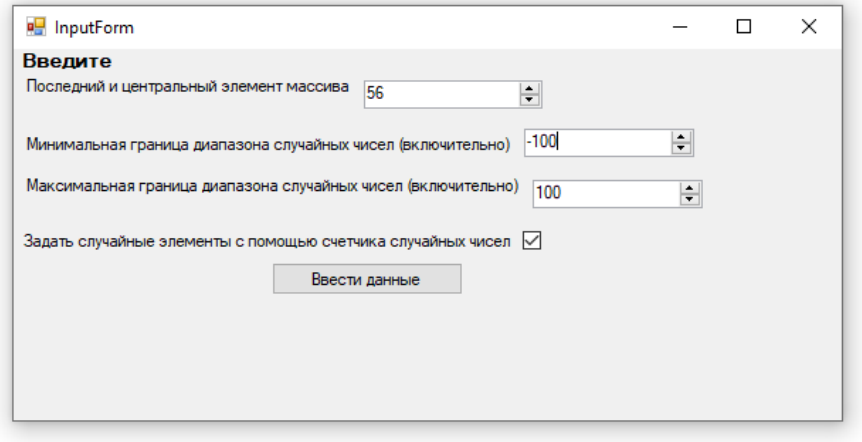

Рисунок 4 – Интерфейс диалогового окна «Input».

После того как пользователь ввел данные, необходимо нажать на пункт меню «Calc» основного окна. После проделанных действий программа откроет диалоговое окно (рис. 5) в котором будет отображен массив который был сгенерирован по параметрам которые ввел пользователь.

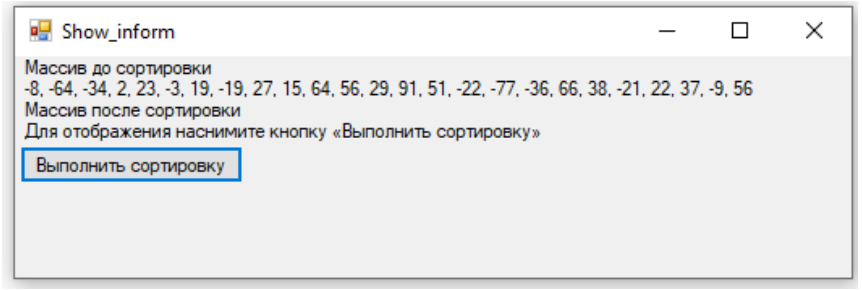

Рисунок 5 – Интерфейс диалогового окна «Calc».

Для того чтобы отсортировать массив по правилам пользователю необходимо нажать на кнопку «Выполнить сортировку» после чего программа отсортирует массив и выведет его на диалоговое окно. На рисунке 6 отображено диалоговое окно с отсортированным массивом.

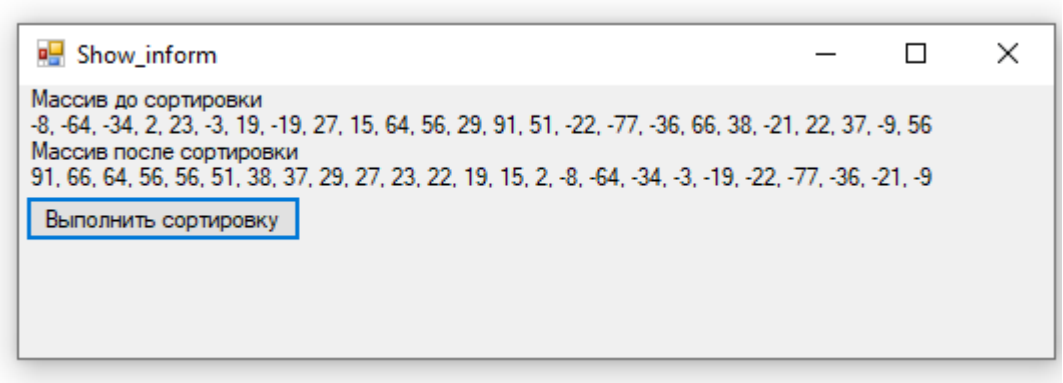

Рисунок 6 – Диалоговое окно с отсортированным массивом.

Для выхода из приложения пользователь может нажать пункт меню «Exit» или закрыть основное окно программы.

Пользователь может не задавать начальные значения в диалоговом окне «Input» и напрямую перейти к выводу значений. В таком случае программа сообщит о ошибке (рис. 7) и попросит пользователя ввести данные.

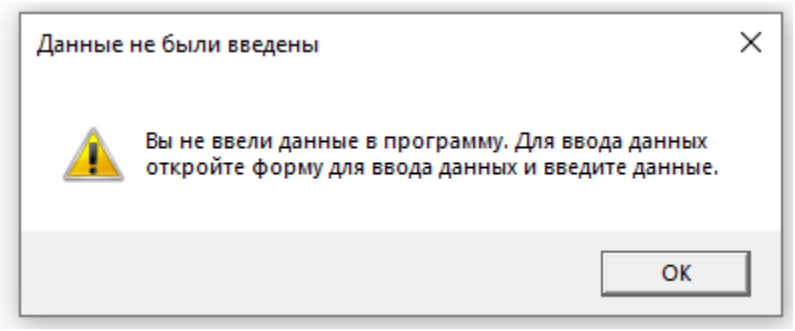

Рисунок 7 – Диалоговое окно сообщающее об отсутствии данных.

ЗАКЛЮЧЕНИЕ

В ходе выполнения практики была изучена литература, посвященная разработке программ с графическим интерфейсом и качестве задания была разработана программа с графическим интерфейсом. Было изучено расширение C++/CLI для языка программирования C++. Данное расширение позволяет создавать программы с использование графического интерфейса.

В ходе разработки программы было отмечено что разработка приложения с интерфейсом на языке C++ с использование расширения C++/CLI позволило получить практический опыт и закрепить ранее полученную теорию. Разработка программы позволила получить новые знания в области создания программ с графическим интерфейсом и интеграции в него консольного кода.

В целом, можно сделать вывод, что практика позволила успешно применить свои знания на практике и развить профессиональные навыки. Все задания, которые давали во время практики, были успешно и своевременно выполнены. Разработка приложения в качестве индивидуального задания была завершена.

Во время прохождения практики все задания были успешно выполнены, и разработка рабочего приложения в качестве индивидуального задания была завершена.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ОБЗОР НОВЫХ ВОЗМОЖНОСТЕЙ VISUAL STUDIO 2022 - Е. А. Пьянова, Е. В. Антонов, О.А. Климов, И.А. Гурин ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина», г. Екатеринбург, Россия

2. Стандартная библиотека C++: справочное руководство – Н. М. Джосаттис

3. C++/CLI: язык Visual C++ для среды .NET – Г. Хогенсон

4. Язык программирования C++. Краткий курс – Б. Страуструп

ПРИЛОЖЕНИЕ А

ТЕКСТ ПРОГРАММЫ

```
// main.cpp
#include "MainWindow.h"
using namespace System;
using namespace System::Windows::Forms;
int main(array<String^>^ args) {
        Application::SetCompatibleTextRenderingDefault(false);
        Application::EnableVisualStyles();
        Program form;
        Application::Run(% form);
        return 0;}
// MainWindow.h
#pragma once
#include <iostream>
#include "InputForm.h"
#include "Show_inform.h"
#include "Exercise.h"
#include "my_arr.h"
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
public ref class Program : public System::Windows::Forms::Form{
public:
   Program(void){InitializeComponent();}
```

```
        protected:
            ~Program(){if (components){delete components;delete my_arrey; }}
        private:
            My_arr<int>* my_arrey = nullptr;
            System::Void exitToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {exit(0);}
            System::Void inputToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e){
                InputForm form_input;
                form_input.ShowDialog(this);
                std::vector<int> res = form_input.get_inform();
                if (res.size()) {delete my_arrey;
                    my_arrey = new My_arr<int>(res);}
                else {MessageBox::Show("Вы не ввели данные в программу. Для
ввода данных вновь откройте форму для ввода данных и введите
данные.","Данные не были введены",
MessageBoxButtons::OK,MessageBoxIcon::Warning,
MessageBoxDefaultButton::Button1,MessageBoxOptions::DefaultDesktopOnly);}
            }
            System::Void calcToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
                if (this->my_arrey == nullptr) {MessageBox::Show("Вы не ввели
данные в программу. Для ввода данных откройте форму для ввода данных и
введите данные.","Данные не были введены",
MessageBoxButtons::OK,MessageBoxIcon::Warning,
MessageBoxDefaultButton::Button1,MessageBoxOptions::DefaultDesktopOnly);
                    return;}
                Show_inform<int> form_show(*(this->my_arrey));
                form_show.ShowDialog(this);}
            System::Void заданиеToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
                Exercise form_exep;
```

```
            form_exep.ShowDialog(this);}
        System::Windows::Forms::MenuStrip^ menuStrip1;
        System::Windows::Forms::ToolStripMenuItem^
inputToolStripMenuItem, calcToolStripMenuItem, exitToolStripMenuItem,
Exercise_menu;
        System::ComponentModel::Container ^components;
        void InitializeComponent(void){
            this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
            this->inputToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->calcToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->exitToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->Exercise_menu = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuStrip1->SuspendLayout();
            this->SuspendLayout();
            this->menuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(4) {this-
>inputToolStripMenuItem,this->calcToolStripMenuItem, this-
>exitToolStripMenuItem, this->Exercise_menu});
            this->menuStrip1->Location = System::Drawing::Point(0, 0);
            this->menuStrip1->Name = L"menuStrip1";
            this->menuStrip1->Size = System::Drawing::Size(284, 24);
            this->menuStrip1->TabIndex = 0;
            this->menuStrip1->Text = L"menuStrip1";
            this->inputToolStripMenuItem->Name = L"inputToolStripMenuItem";
            this->inputToolStripMenuItem->Size = System::Drawing::Size(47, 20);
            this->inputToolStripMenuItem->Text = L"Input";
```

```
            this->inputToolStripMenuItem->Click += gcnew
System::EventHandler(this, &Program::inputToolStripMenuItem_Click);
            this->calcToolStripMenuItem->Name = L"calcToolStripMenuItem";
            this->calcToolStripMenuItem->Size = System::Drawing::Size(42, 20);
            this->calcToolStripMenuItem->Text = L"Calc";
            this->calcToolStripMenuItem->Click += gcnew
System::EventHandler(this, &Program::calcToolStripMenuItem_Click);
            this->exitToolStripMenuItem->Name = L"exitToolStripMenuItem";
            this->exitToolStripMenuItem->Size = System::Drawing::Size(38, 20);
            this->exitToolStripMenuItem->Text = L"Exit";
            this->exitToolStripMenuItem->Click += gcnew
System::EventHandler(this, &Program::exitToolStripMenuItem_Click);
            this->Exercise_menu->Name = L"Exercise_menu";
            this->Exercise_menu->Size = System::Drawing::Size(64, 20);
            this->Exercise_menu->Text = L"Задание";
            this->Exercise_menu->Click += gcnew System::EventHandler(this,
&Program::заданиеToolStripMenuItem_Click);
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
            this->BackColor = System::Drawing::SystemColors::ButtonHighlight;
            this->ClientSize = System::Drawing::Size(284, 161);
            this->Controls->Add(this->menuStrip1);
            this->MainMenuStrip = this->menuStrip1;
            this->MinimumSize = System::Drawing::Size(300, 200);
            this->Name = L"Program";
            this->Text = L"Буханов Денис Евгеньевич, группа 3105 об, 3
вариант";
            this->menuStrip1->ResumeLayout(false);
            this->menuStrip1->PerformLayout();
```

```
            this->ResumeLayout(false);
            this->PerformLayout();}};
        // InputForm.h
        #pragma once
        #include <iostream>
        #include <ctime>
        #include <vector>
        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;
        namespace my_random {
           int min_range = 0;
           int max_range = 100;
           void set_range(int min_range, int max_range) {
              if (min_range >= max_range) {throw std::invalid_argument("Верхняя
граница для диапазона рандомных чисел должна быть больше нижней
границы.");}
              my_random::min_range = min_range;
              my_random::max_range = max_range;}
           void srand(unsigned int option = ((time(NULL)) % 1000))
{std::srand(option);}
           int random() {return ((rand() % (max_range - min_range + 1)) +
min_range);}}
        public ref class InputForm : public System::Windows::Forms::Form{
        public:
           InputForm(void){InitializeComponent();}
           std::vector<int> get_inform(){
```

```
            if (this->state_inform) {
                std::vector<int> items(25);
                if (this->use_random->Checked) {
                    for (int i = 0; i < items.size(); i += 1) {
                        items[i] = my_random::random();}}
                items[11] = int(this->super_item->Value);
                items[24] = int(this->super_item->Value);
                return (items);}
            std::vector<int> items(0);
            return (items);}
        protected:
            ~InputForm(){if (components){delete components;}}
        private:
            bool state_inform = false;
            System::Void set_inform_Click(System::Object^ sender,
System::EventArgs^ e){
                try{
                    my_random::srand();
                    my_random::set_range(int(this->min_range_random->Value),
int(this->max_rande_random->Value));
                    this->state_inform = true; }
                catch (std::invalid_argument){
                    MessageBox::Show("Верхняя граница для диапазона рандомных
чисел должна быть больше нижней границы.","Введены не верные значения",
MessageBoxButtons::OK,MessageBoxIcon::Error,
MessageBoxDefaultButton::Button1,MessageBoxOptions::DefaultDesktopOnly);}
}
            System::Windows::Forms::Label^ label2, label3, label4, label5;
            System::Windows::Forms::NumericUpDown^ super_item,
min_range_random, max_rande_random;
```

```
        System::Windows::Forms::FlowLayoutPanel^ flowLayoutPanel1,
flowLayoutPanel2, flowLayoutPanel3, flowLayoutPanel4;

        System::Windows::Forms::CheckBox^ use_random;

        System::Windows::Forms::Button^ set_inform;

        System::ComponentModel::Container^ components;

        void InitializeComponent(void){

            this->label2 = (gcnew System::Windows::Forms::Label());

            this->label3 = (gcnew System::Windows::Forms::Label());

            this->super_item = (gcnew
System::Windows::Forms::NumericUpDown());

            this->min_range_random = (gcnew
System::Windows::Forms::NumericUpDown());

            this->max_rande_random = (gcnew
System::Windows::Forms::NumericUpDown());

            this->label4 = (gcnew System::Windows::Forms::Label());

            this->label5 = (gcnew System::Windows::Forms::Label());

            this->flowLayoutPanel1 = (gcnew
System::Windows::Forms::FlowLayoutPanel());

            this->flowLayoutPanel2 = (gcnew
System::Windows::Forms::FlowLayoutPanel());

            this->flowLayoutPanel3 = (gcnew
System::Windows::Forms::FlowLayoutPanel());

            this->flowLayoutPanel4 = (gcnew
System::Windows::Forms::FlowLayoutPanel());

            this->use_random = (gcnew System::Windows::Forms::CheckBox());

            this->set_inform = (gcnew System::Windows::Forms::Button());

            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>super_item))->BeginInit();

            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>min_range_random))->BeginInit();

            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>max_rande_random))->BeginInit();};
```