

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Рязанский государственный радиотехнический университет им. В.Ф. Уткина
Кафедра электронных вычислительных машин

К ЗАЩИТЕ
руководитель КП
_____ А.С. Тарасов
« ____ » _____ 2023 г.

КУРСОВАЯ РАБОТА
по дисциплине
«Основы алгоритмизации и ООП»
на тему
«Разработка приложения
с использованием динамических структур данных»

Выполнил студент группы 247
Деев Е. В.

дата сдачи на проверку, подпись

Руководитель проекта
Асс. каф. ЭВМ Тарасов А.С.

оценка

дата защиты, подпись

Рязань 2023 г.

ЗАДАНИЕ
на курсовую
работу по
дисциплине
«Основы алгоритмизации и ООП»

Студента Деева Е. В., гр. 247

- 1. Тема работы:** «Разработка приложения с использованием динамических структур данных»
- 2. Срок сдачи студентом законченной работы**
- 3. Руководитель работы:** *Тарасов Андрей Сергеевич, ассистент кафедры ЭВМ РГРТУ.*
- 4. Исходные данные к работе**

- 1. Операционная система Windows XP/7/8/10*
- 2. Язык программирования C/C++*
- 3. Среда программирования MS Visual Studio C++*
- 4. Индивидуальное задание*

5. Содержание пояснительной записки

Задание

Содержание

Введение

1. Постановка задачи

2. Разработка программы

3. Руководство оператора

Заключение

Список использованных источников

Приложение – листинги программ

Задание выдано «15» марта 2023 г. _____ / Асс. Каф. ЭВМ. Тарасов А.С.

**Индивидуальное задание к курсовой работе
по дисциплине «Основы алгоритмизации и ООП»**

студента группы 247 Деева Е. В.

Общие требования

Вариант задания связан с разработкой таблицы данных с использованием линейных однонаправленных списков.

Разрабатываемая программа должна обязательно выполнять следующие запросы:

- заполнение пустой таблицы;
- сохранение таблицы в файле;
- чтение таблицы из файла;
- вывод таблицы на экран;
- добавление элементов в таблицу;
- редактирование элементов таблицы;
- удаление элементов из таблицы;
- а также все запросы, которые указаны в индивидуальном задании.

Вызовы запросов должны осуществляться через систему меню с использованием средств визуального программирования *Visual Studio (Visual C++)*. Необходимо предусмотреть контроль ошибок пользователя при вводе данных. Результаты некоторых запросов должны выводиться в виде графиков или диаграмм.

При запуске приложение должно выдавать заставку, которая отражает назначение приложения.

Все элементарные действия должны быть оформлены в виде подпрограмм, а некоторые объявления и подпрограммы должны быть оформлены в виде модуля (модулей).

Индивидуальное задание

Вариант № 3. Анкета абитуриента

Анкета абитуриента имеет следующие пункты:

- ФИО;
- адрес постоянного проживания;
- льгота (инвалидность, сирота, целевой набор);
- баллы по ЕГЭ (математика, физика, русский язык);
- номер направления. По каждой специальности определен проходной балл.

Написать программу, которая выполняет следующие запросы:

- упорядочение таблицы по ФИО абитуриента;
- определение процента иногородних абитуриентов;
- вывод списка абитуриентов, которые поступили на определенное направление;
- вывод номеров направлений в порядке убывания «популярности»;
- вывод процентного соотношения льготников и абитуриентов, которые поступают в вуз на общих основаниях.

Содержание

Введение.....	6
1. Постановка задачи.....	8
2. Разработка программы.....	9
2.1 Руководство оператора.....	11
2.2 Сообщения оператору.....	12
3. Заключение.....	13
Список использованных источников.....	14
Приложение А.....	15
Приложение Б.....	16

Введение

Курсовая работа (КР) представляет собой самостоятельную работу по теме **«Разработка приложения с использованием динамических структур данных»**.

Работа предполагает:

- домашнюю внеаудиторную подготовку;
- консультации по КР;
- предъявление промежуточных результатов для проверки и контроля выполнения курсовой работы;
- написание и оформление пояснительной записки;
- сдачу и защиту КР в сроки согласно учебному графику.

При выполнении данной работы использовались динамические структуры данных, как и указано в теме. Под динамической структурой данных понимается любая структура данных, занимаемый объем памяти которой не является фиксированным. Существует несколько разновидностей динамических структур: список, стек, очереди и т.д. Различие состоит в характере связи компонентов между собой. Каждый элемент (узел) состоит из двух областей памяти: поля данных и ссылок. Ссылки – это адреса других узлов этого же типа, с которыми данный элемент логически связан. В языке Си для организации ссылок используются переменные указатели. При добавлении нового узла в такую структуру выделяется новый блок памяти и (с помощью ссылок) устанавливаются связи этого элемента с уже существующими.

Одной из используемых структур такого типа является список. Список - динамическая структура данных, добавление элементов в которую или удаление существующих возможно в любом месте. По организации связи различают: однонаправленные, двунаправленные и кольцевые.

Также при выполнении работы использовались классы. Под классами понимается абстрактный тип данных. Он сочетает в себе два функционала:

Первая — это структура, в которой можно хранить различные типы данных: массивы, переменные, функции.

Вторая — возможность пользоваться объектно-ориентированным программированием.

Создав класс можно создать его экземпляр — объект.

При разработке приложения активно использовалась **техника многомодульного программирования**. Она заключается в использовании различных модулей, содержащих различные части кода, необходимые для его отлаженной работы. Многомодульное программирование также способствует более удобоваримому и наглядному представлению кода.

Основная часть

1. Постановка задачи

Целью курсовой работы является разработка приложения, написанного на алгоритмическом языке программирования C++ в среде визуального программирования Visual Studio C++.

Разработанная программа должна выполнять следующие функции:

- упорядочение таблицы по ФИО абитуриента;
- определение процента иногородних абитуриентов;
- вывод списка абитуриентов, которые поступили на определенное направление;
- вывод номеров направлений в порядке убывания «популярности»;
- вывод процентного соотношения льготников и абитуриентов, которые поступают в вуз на общих основаниях.

Для разработки приложения будем использовать Windows Form и ее элементы, необходимые для корректной работы пользователя с целью получения требуемой информации и успешной передачи собственной.

Основную часть занимает место для вывода нашей таблицы с данными. Таблица расположена в самом центре слева, чтобы информация – самое важное в работе программы – привлекала внимание пользователя, а сбоку кнопки реализованных функций, выпадающий список с номерами направлений и четверть экрана занимает диаграмма, показывающая соотношение льготников и нет.

В самом верху располагается небольшое меню, в котором можно обновить таблицу и добавить нового абитуриента.

2. Разработка программы

Для разработки приложения был выбран высокоуровневый компилируемый язык программирования C++. Язык C++ позволяет создавать приложения в объектно-ориентированном стиле, то есть представлять его как взаимодействующие между собой методы и объекты.

Данный язык отличается от иных высокоуровневых языков программирования скоростью работы, что невероятно важно при создании крупных программ различной направленности. Таким образом, выбор данного языка обусловлен требованием к наиболее быстрому исполнению программы ради удобства работы.

Также, важно отметить, что при создании программы был использован интерфейс **Windows Form**. Именно он обуславливает работу пользователя с созданной программой. Интерфейс использует управляемый код, исполняющий различные действия кнопок/полей ввода/таймеров при помощи классов управления **System.WinForms**. Это упрощает программирование для начинающего программиста, потому выбор интерфейса оправдан.

При обдумывании подхода программирования было принято решение об использовании техники многомодульного программирования. Код был разбит на несколько модулей – исполняющий код основных функций был описан в `MyForm.h`. В модуле `config.h` была описана структура с элементами «surname», «name», «patro» - ФИО абонента, «address» - адрес, «benefit» - льгота, «score_UGE» - баллы по ЕГЭ, «trend» - номер направления.

Файл «InvestWinApp.cpp»:

- функция «writeInFile» записывает список в текстовый файл, получая уже записанные данные, добавляя к ним нового абонента.
- функция «MarshalString» представлена в официальной документации Майкрософт, как метод интерпритации переменной из типа «System::String^» в «std::string», для последующего использования в коде.

Файл «AddApplicant.h»:

- функция «btn_add_Click» сохраняет введенные данные о пользователе в файл с помощью функции «writeInFile». Перед этим смотря, если данные были введены верно, то они присваиваются переменной списка и эта переменная передается в функцию «writeInFile». Иначе выводится окно с уведомлением об ошибке ввода. После поля очищаются, и форма закрывается.

Файл «MyForm.h»:

- функция «Chart» получает список абонентов из файла. В цикле пробегает по всем абонентам и определяет, если льгота есть, то счетчик количества людей со льготой увеличивается на 1, иначе счетчик количества студентов на общих основаниях увеличивается на 1. После инициализируем диаграмму, высчитываем процентное соотношение преподавателей и студентов и передаём его в диаграмму.
- функция «Table» добавляет новые строки в таблице, превращая данные в удобоваримый вид.
- функции «добавитьАбитуриентаToolStripMenuItem_Click» вызывает форму по добавлению абитуриента.
- функции «after_btn_Click» является кнопкой, выводящей MessageBox с процентом иногородних студентов.
- функции «top_btn_Click» является кнопкой, выводящей MessageBox со списком номеров направлений по их популярности (кол-во упоминаний в таблице).

Полный листинг описанных функций представлен в Приложении Б.

Руководство оператора

Программа предназначена для работы с информацией о абитуриентах. Добавление абитуриентов, вывод информации о абитуриентах, вывод популярных направлений, процента иногородних студентов и соотношение льготников к не льготникам.

Условия выполнения программы:

Установленная Windows 7/8/10/11 64 bit. Дисковое пространство, не менее 1 МБ, которое может вырасти по мере добавления информации.

Выполнение программы:

Двойным щелчком в проводнике запустить программу. (Приложение А, рисунок 1)

Для создания строки таблицы необходимо нажать на пункт меню «Таблица» и выбрать «Добавить Абитуриента», после чего выведется дополнительная форма для внесения информации об абоненте. Если какие-то поля оставить пустыми, то выведется предупреждение. (Приложение А, рисунок 2)

Все остальные данные и взаимодействия находятся на основном экране и будут обновляться по мере работы с программой автоматически.

Сообщения оператору:

«Некорректный ввод!» - пользователь ввёл не полную информацию об абоненте или ввёл некорректно. Для исправления данной ошибки следует заполнить пустые поля для ввода.

«Вывод» - при демонстрации списка направлений по популярности и процента иногородних студентов.

Заключение

В ходе выполнения данной курсовой работы была разработана программа для работы с информацией об абонентах библиотеки с использованием таблицы данных на основе класса и получены практические навыки разработки программ с графическим интерфейсом.

Цель работы была успешно достигнута. Были выполнены все поставленные задачи. Программа позволяет выполнять различные действия с данными об абонентах библиотеки. Есть графический интерфейс для более удобного и интуитивного взаимодействия с программой.

Список использованных источников

1. Хабр URL: <https://habr.com/>
2. Microsoft C++ Documentation URL: <https://docs.microsoft.com/>
3. GitHub URL: <https://github.com/>
4. StackOverflow URL: <https://stackoverflow.com/>
5. Википедия URL: <https://ru.wikipedia.org/>
6. StudFiles URL: <https://studfile.net/>
7. СДО РГРТУ URL: <https://cdo.rsreu.ru/>

Приложение А.

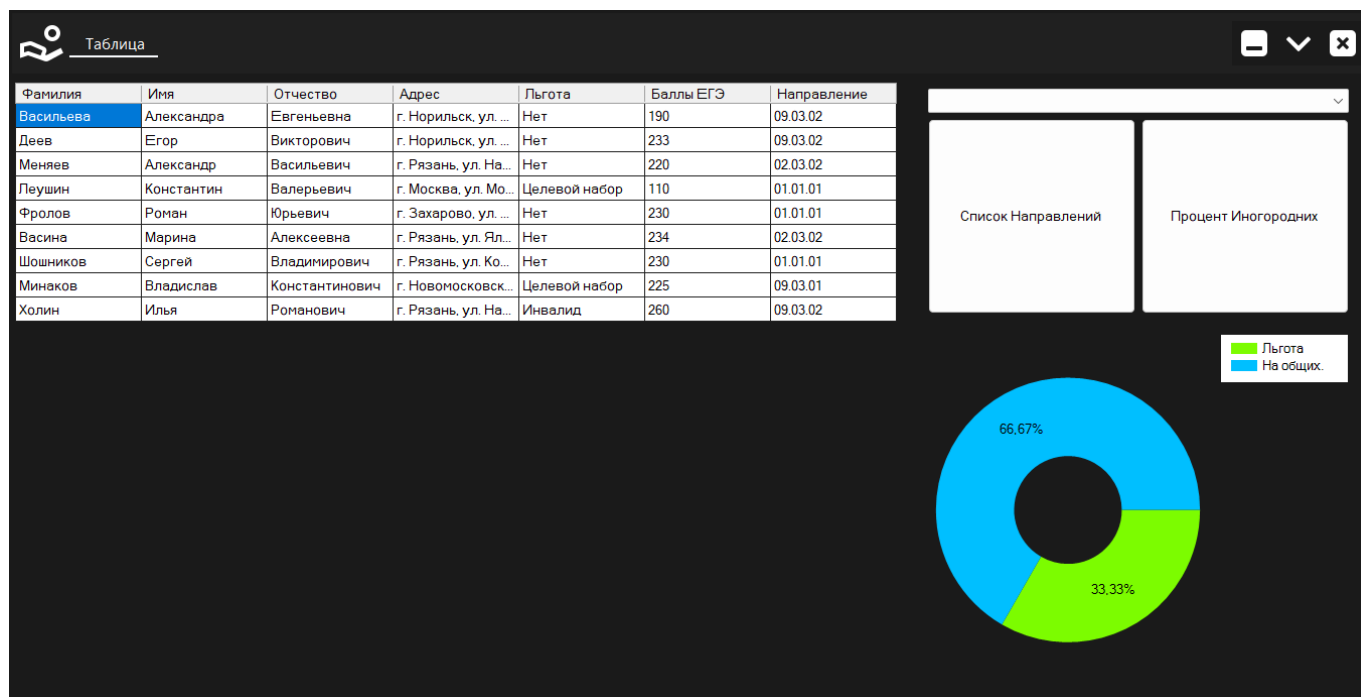


Рисунок 1 – Вывод таблицы

The screenshot shows a web form titled 'AddApplicant'. It contains input fields for 'Фамилия', 'Имя', and 'Отчество' (grouped), 'Адрес', 'Льгота' (with a radio button and the text 'Нет'), 'Баллы по ЕГЭ', and 'Номер направления'. A large 'Добавить' button is at the bottom.

AddApplicant

Фамилия Имя Отчество

Адрес

Льгота ☐ Нет

Баллы по ЕГЭ

Номер направления

Добавить

Рисунок 2 – Внесение информации об абоненте

Приложение Б.

config.h

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <fstream>
#include <string>
#include <iomanip>
#include <map>
#include <set>

using namespace RSREU::IO;
using namespace System;
using namespace System::Windows::Forms;
using namespace System::Collections::Generic;

[Serializable]
ref struct Applicant {
    String^ surname;
    String^ name;
    String^ patro;

    String^ address;

    String^ benefit;
    UInt32 score_UGE;
    String^ trend;
};

void writeInFile(List<Applicant^>^ list);
void MarshalString(String^ s, std::string& os);
```


MyForm.h

```
// отслеживание действий формы
private: System::Void MyForm_Load(System::Object^ sender, System::EventArgs^ e) {
    main_ico_Click(sender, e);
}
private: System::Void MyForm_Closed(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void MyForm_MouseMove(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e) {
    if (e->Button == System::Windows::Forms::MouseButtons::Left) {
        this->ClientSize = System::Drawing::Size(this->PointToClient(MousePosition).X,
this->PointToClient(MousePosition).Y);
    }
}

// кнопки взаимодействия с окном
private: System::Void close_Click(System::Object^ sender, System::EventArgs^ e) {
    this->Close();
}
private: System::Void expand_Click(System::Object^ sender, System::EventArgs^ e) {
    this->TopMost = true;
    if (!WindowWrap) {
        this->WindowState = System::Windows::Forms::FormWindowState::Maximized;
WindowWrap = 1;
    } else {
        this->WindowState = System::Windows::Forms::FormWindowState::Normal;
WindowWrap = 0;
    }
}
private: System::Void wrap_Click(System::Object^ sender, System::EventArgs^ e) {
    this->TopMost = true;
    this->WindowState = System::Windows::Forms::FormWindowState::Minimized;
}

// передвижения окна по экрану
private: System::Void moveWindow_MouseDown(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e) {
    mPoint = Point(e->X, e->Y);
}
private: System::Void moveWindow_MouseMove(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e) {
    if (e->Button == System::Windows::Forms::MouseButtons::Left) {
        this->Location = Point(this->Location.X + e->X - mPoint.X, this->Location.Y +
e->Y - mPoint.Y);
    }
}

// обновление таблицы
private: System::Void main_ico_Click(System::Object^ sender, System::EventArgs^ e) {
    Table(sender, e);
}

private: System::Void after_btn_Click(System::Object^ sender, System::EventArgs^ e) {
    List<Applicant>^ data = ReadWrite::Load<List<Applicant>^>("applis.dat");

    if (data) {
        int local = 0; int unlocal = 0;
        for (int i = 0; i < data->Count; i++) {
            std::string adr = "";
            MarshalString(data[i]->address, adr);
            adr = adr.substr(0, adr.find(", "));

            if (adr == "г. Рязань") local++;
            else unlocal++;
        }
    }
}
```

```

    }

    double perc = (100 / ((double)data->Count)) * unlocal;
    MessageBox::Show("Процент иногородних абитуриентов составил " +
perc.ToString()->Substring(0, 4) + "%", "Вывод");
    }
}

private: System::Void top_btn_Click(System::Object^ sender, System::EventArgs^ e) {
    List<Applicant^>^ data = ReadWrite::Load<List<Applicant^>^>("applis.dat");
    List<String^>^ nums = gcnew List<String^>;

    if (data) {
        for (int i = 0; i < data->Count; i++)
            if (!nums->Contains(data[i]->trend)) nums->Add(data[i]->trend);

        int* nums_i = new int[nums->Count];
        for (int i = 0; i < nums->Count; i++) nums_i[i] = 0;

        for (int i = 0; i < data->Count; i++)
            for (int j = 0; j < nums->Count; j++)
                if (nums[j] == data[i]->trend) nums_i[j]++;

        for (int i = 0; i < nums->Count; i++) {
            for (int j = 0; j < nums->Count - 1; j++) {
                if (nums_i[j] < nums_i[j + 1]) {
                    std::swap(nums_i[j], nums_i[j + 1]);

                    String^ buf = nums[j];
                    nums[j] = nums[j + 1];
                    nums[j + 1] = buf;

                }
            }
        }

        String^ txt = "Перечень направлений по популярности:\n";
        for (int i = 0; i < nums->Count; i++) txt += (i + 1).ToString() + ". " +
nums[i] + " в кол-ве " + nums_i[i].ToString() + "\n";

        MessageBox::Show(txt, "Вывод");
    }
}

private: System::Void comboBox_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e) {
    Table(sender, e);
    Chart(sender, e);
}

// самописные функции
private: System::Void Table(System::Object^ sender, System::EventArgs^ e) {
    gridTable->Rows->Clear();
    List<Applicant^>^ data = ReadWrite::Load<List<Applicant^>^>("applis.dat");
    List<String^>^ nums = gcnew List<String^>;

    if (data) {
        for (int i = 0; i < data->Count; i++) {
            if (comboBox->Text == "" || comboBox->Text == data[i]->trend)
                gridTable->Rows->Add(data[i]->surname, data[i]->name, data[i]-
>patro,
                    data[i]->address, data[i]->benefit, data[i]->score_UGE,
                    data[i]->trend);

            if (!nums->Contains(data[i]->trend)) nums->Add(data[i]->trend);
        }
    }
}

```

```

    }

    comboBox->Items->Clear();
    for (int i = 0; i < nums->Count; i++) {
        comboBox->Items->Add(nums[i]);
    }
}

Chart(sender, e);
}

private: System::Void Chart(System::Object^ sender, System::EventArgs^ e) {
    List<Applicant^>^ data = ReadWrite::Load<List<Applicant^>^>("applis.dat");

    if (data) {
        int bYes = 0, bNo = 0;

        for (int i = 0; i < data->Count; i++) {
            if (data[i]->benefit == "Нет") bNo++;
            else bYes++;
        }
        double proc = 100 / (double)data->Count;

        auto s1 = chart->Series[0];
        s1->Points->Clear();

        s1->Points->AddY(bYes * proc);
        s1->Points[0]->Color = System::Drawing::Color::LawnGreen;
        s1->Points[0]->Label = L"#PERCENT{P2}";
        s1->Points[0]->LegendText = L"Льгота";

        s1->Points->AddY(bNo * proc);
        s1->Points[1]->Color = System::Drawing::Color::DeepSkyBlue;
        s1->Points[1]->Label = L"#PERCENT{P2}";
        s1->Points[1]->LegendText = L"На общих.";
    }
}

// функции меню
private: System::Void обновитьТаблицуToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
    Table(sender, e);
    MessageBox::Show("Таблица успешно обновлена!", "Таблица");
}

private: System::Void добавитьАбитуриентаToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
    AddApplicant^ prt_data = gcnew AddApplicant();
    prt_data->ShowDialog();

    Table(sender, e);
}

```

AddApplicant.h

```
private: System::Void btn_add_Click(System::Object^ sender, System::EventArgs^ e) {
    // вычисление id абонента
    List<Applicant^>^ data = ReadWrite::Load<List<Applicant^>>("applis.dat");

    // получение данных
    List<Applicant^>^ abit = gcnew List<Applicant^>;
    abit->Add(gcnew Applicant);

    bool flag = true;
    try {
        abit[0]->surname = surnameBox->Text;
        abit[0]->name = nameBox->Text;
        abit[0]->patro = parBox->Text;

        abit[0]->address = addressBox->Text;

        if (benefitNo->Checked) abit[0]->benefit = "Нет";
        else abit[0]->benefit = benefitYes->Text;

        abit[0]->score_UGE = Convert::ToInt32(scoreBox->Text);
        abit[0]->trend = trendBox->Text;

        // запись в файл
        writeInFile(abit);
    }
    catch (...) { MessageBox::Show("Некорректный ввод!"); flag = false; }

    // очистка формы
    if (flag) {
        surnameBox->Clear();
        nameBox->Clear();
        parBox->Clear();
        addressBox->Clear();
        benefitNo->Checked = false;
        benefitYes->Clear();
        scoreBox->Clear();
        trendBox->Clear();
    }
}

private: System::Void benefitNo_Click(System::Object^ sender, System::EventArgs^ e) {
    benefitYes->Clear();
    benefitNo->Checked = true;
}

private: System::Void benefitYes_TextChanged(System::Object^ sender, System::EventArgs^ e)
{
    benefitNo->Checked = false;
}
```

BiblioAbonents.cpp

```
#include "pch.h"

#include "config.h"
#include "AddApplicant.h"
#include "MyForm.h"

void writeInFile(List<Applicant^>^ list) {
    List<Applicant^>^ old_data = ReadWrite::Load<List<Applicant^>^>("applis.dat");

    if (old_data) {
        old_data->AddRange(list);
        ReadWrite::Save(old_data, "applis.dat");
    }
    else ReadWrite::Save(list, "applis.dat");
}

void MarshalString(String^ s, std::string& os) {
    using namespace Runtime::InteropServices;
    const char* chars = (const char*)(Marshal::StringToHGlobalAnsi(s)).ToPointer();
    os = chars;
    Marshal::FreeHGlobal(IntPtr((void*)chars));
}

[STAThreadAttribute]
int main(array<String^>^ args) {
    Application::SetCompatibleTextRenderingDefault(false);
    Application::EnableVisualStyles();
    InvestWinApp::MyForm form;
    Application::Run(% form);
}
```