МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»

Факультет физико-математических и естественных наук

Кафедра информационных технологий

«УТВЕРЖДАЮ»
Заведующий кафедрой
информационных технологий
д.ф.-м.н., проф.

_____Ю.Н. Орлов

«___»_____ 2023г.

ОТЧЕТ
по лабораторной работе №1

по дисциплине «Технология программирования»

Выполнил:

Студент группы НПИбд-02-22

Студенческий билет № 1132226528

Тарутина Кристина Олеговна

_____
(Подпись)

«22» Февраля 2023г.

Москва 2023

Цель

Написать компьютерную программу, содержащую

- Описание структуры, содержащей поля типа string, int, double;
- Набор функций для работы со списком на базе этой структуры:
  - Добавление элемента в начало списка;
  - Добавление элемента в конец списка;
  - Добавление элемента в список после заданного элемента;
  - Добавление элемента в список перед заданным элементом;
  - Удаление из списка элемента с заданным именем;
  - Вывод содержания списка на экран;
- Функцию main, содержащую сценарий работы со списком, использующий разработанный инструментарий.

Ход лабораторной работы

Шаг 1: Описание структуры

Сначала я описываю структуру для списка студентов РУДН, где будут храниться имя, фамилия, номер студенческого билета, количество баллов, а также оценки в русском и английском формате(рис 1)

```
struct Student{ //Описание структуры, содержащей поля типа string, int, char, double
string name{};
string surname{};
char english_grade{};
int student_ID{};
int russian_grade{};
double points{};
Student *next{};
};
```

Рис 1

Шаг 2: Создание функций

После этого я создаю функцию для добавления элемента в начало списка. Сперва я объявляю указатель для описанной мной ранее структуры, и заполняю поля из структуры значениями, которые передаю при вызове функции. В поле next я вкладываю значение головы списка, после чего заменяю значение головы на нынешний элемент.(рис 2)

```
void add_to_start(Student *& Head,string name, string surname, int student_ID, double points,
                  int russian_grade, char english_grade){ //Добавление элемента в начало списка
    Student *newstudent = new Student;
    newstudent->name = name;
    newstudent->surname = surname;
    newstudent->student_ID = student_ID;
    newstudent->points = points;
    newstudent->russian_grade = russian_grade;
    newstudent->english_grade = english_grade;
    newstudent->next = Head;
    Head = newstudent;
    return;
}
```

Рис 2

Следующей я создаю функцию для добавления элемента в конец списка. Здесь я объявляю указатель временной переменной, внутрь которого вкладываю значение "головы" списка. Сразу проверяю не равна ли голова списка нулю, ибо тогда можно просто вызвать функцию добавления элемента в начало списка и не утруждаться далее.

Если же голова списка не равна нулю, то в цикле перемещаемся по списку с помощью поля next, после чего повторяем действия из прошлого цикла, также создавая указатель для описанной структуры и заполняя поля, за одним небольшим исключением. Теперь в поле next у нас будет храниться значение NULL, так как после на данный момент следующей структуры нет.

```
void add_to_end(Student *& Head,string name, string surname, int student_ID, double points,
                int russian_grade, char english_grade){ //Добавление элемента в конец списка;
    Student *tmp = Head;
    if (Head == NULL){
        add_to_start( & Head, name, surname, student_ID, points, russian_grade, english_grade);
        return;
    }
    while (tmp->next) tmp = tmp->next;
    Student *newstudent = new Student;
    newstudent->name = name;
    newstudent->surname = surname;
    newstudent->student_ID = student_ID;
    newstudent->points = points;
    newstudent->russian_grade = russian_grade;
    newstudent->english_grade = english_grade;
    newstudent->next = NULL;
    tmp->next = newstudent;
    return;
}
```

Рис 3

Следующим реализуем функцию вставки элемента после заданного. Для этого помимо привычного набора данных, мы передаём в функцию также номер студенческого билета студента, после которого в список мы хотим добавить нового студента.

Сразу проверяем, чтобы список не был пустым, сверяясь не лежит ли в значении головы списка NULL и объявляем указатель, после чего в цикле, до тех пор, пока мы не дойдём до конца или пока не найдём нужного студента по номеру, мы переходим на следующий элемент списка. Снова проверяем не дошли ли мы до конца списка, и если нет, то добавляем в список нового студента. (рис 4)

```
int insert_after(Student *& Head, string name, string surname, int student_ID, double points,
                 int russian_grade, char english_grade, int student_ID_After){ //Добавление элемента в список
    // после заданного элемента
    if (Head == NULL) return 1;
    Student *tmp = Head;
    while (tmp != NULL && tmp -> student_ID != student_ID_After)
        tmp = tmp->next;
    if (tmp == NULL) return 1;
    Student *newstudent = new Student;
    newstudent->name = name;
    newstudent->surname = surname;
    newstudent->student_ID = student_ID;
    newstudent->points = points;
    newstudent->russian_grade = russian_grade;
    newstudent->english_grade = english_grade;
    newstudent->next = tmp->next;
    tmp->next = newstudent;
    return 0;
}
```

Рис 4

Теперь приступаем к функции вставки до указанного элемента. Во многом эта функция похожа не предыдущую, за небольшим исключением. Теперь мы также в начале проверяем не только, что список пустой, но и не является ли нужный нам студент началом данного списка. Если это действительно так, то мы просто пользуемся функцией вставки элемента в начало списка. После этого мы объявляем два указателя, один на голову списка, другой на последующий за головой элемент. Цикл почти такой же, как в предыдущей программе, за исключением того, что передвигаем мы не один, а два указателя. (рис 5)

```
int insert_before(Student *& Head, string name, string surname, int student_ID, double points,
                  int russian_grade, char english_grade, int student_ID_Before){ //Добавление элемента в список
    // перед заданным элементом;
    if (Head == NULL) return 1;
    if (Head->student_ID == student_ID_Before){
        add_to_start( & Head, name, surname, student_ID, points, russian_grade, english_grade);
        return 0;
    }
    Student *prev = Head, *tmp=Head->next;
    while (tmp != NULL && tmp -> student_ID != student_ID_Before){
        prev = prev->next;
        tmp = tmp->next;
    }
    if (tmp->next == NULL && tmp->student_ID!=student_ID_Before) return 1;
    Student *newstudent = new Student;
    newstudent->name = name;
    newstudent->surname = surname;
    newstudent->student_ID = student_ID;
    newstudent->points = points;
    newstudent->russian_grade = russian_grade;
    newstudent->english_grade = english_grade;
    newstudent->next = tmp->next;
    tmp->next = newstudent;
    return 0;
}
```

Рис 5

Для функции по удалению предмета нам нужно лишь знать голову списка и номер студенческого билета того, кого мы хотим удалить.

Сразу же проверяем, что наш список не пустой, после чего объявляем указатель и, если нужный студент в начале списка, удаляем его, переменной Head назначая следующий элемент.

Если студент не в начале списка, то объявляем ещё один указатель. Он будет содержать голову списка, в то время как предыдущий указатель последующий элемент.

Снова реализуем цикл поиска студента по номеру его студенческого билета (рис 6)

```
int remove_from_list(Student *& Head, int student_ID){ //Удаление из списка элемента с заданным именем
    if (Head == NULL) return 1;
    Student *tmp = Head;
    if (Head->student_ID == student_ID){
        Head = Head->next;
        delete tmp;
        return 0;
    }
    if (Head->next == NULL) return 1;
    Student *prev = Head;
    tmp = Head->next;
    while (tmp->next != NULL && tmp->student_ID != student_ID){
        prev = prev->next;
        tmp = tmp->next;
    }
    if (tmp->next==NULL && tmp->student_ID != student_ID)
        return 1;
    prev -> next = tmp -> next;
    delete tmp;
    return 0;
}
```

Рис 6

Последней создаём функцию вывода списка на экран. Проверяем, что список не пуст, и если нет, то уже использованным до этого методом проходимся по всем элементам списка до самого конца.(рис 7)

```
void print(Student *Head){ // Вывод содержания списка на экран;
    int i = 1;
    if (Head == NULL){
        cout<<"Список пуст";
        return;
    }
    Student *tmp = Head;
    while (tmp != NULL){
        cout << i << ") " << tmp->name << ' ' << tmp->surname << ' ' << tmp->student_ID << ' ';
        cout << tmp->points << ' ' << tmp->russian_grade << ' ';
        cout << tmp->english_grade << "; " << endl;
        i++;
        tmp = tmp->next;
    }
    return;
}
```

Рис 7

Шаг 3: Функция main

С помощью условных операторов, цикла while, а также операций ввода/вывода, я реализую возможность пользователю самому создавать, изменять и выводить список студентов, а также удалять конкретных лиц по желанию(рис 8-10)

```
int main(){
    setlocale( Category: LC_ALL,  Locale: "Russian");
    typedef Student *students;
    students Head = NULL;
    int n, student_ID, russian_grade, student_ID_before, student_ID_after;
    string name, surname;
    char english_grade;
    double points;
    cout << "Краткое руководство по работе с данной программой: " << endl;
    cout << "Данная программа позволяет хранить список отметок студентов, а также самые базовые их ";
    cout << "данные, такие как имя и фамилия, а также номер студенческого билета";
    cout << "Если вы хотите добавить студента в начало списка - нажмите 1" << endl;
    cout << "Если вы хотите добавить студента в конец списка - нажмите 2" << endl;
    cout << "Если вы хотите добавить студента после какого-то другого студента - нажмите 3" << endl;
    cout << "Если вы хотите добавить студента до какого-то другого студента - нажмите 4" << endl;
    cout << "Если вы хотите удалить студента из списка - нажмите 5" << endl;
    cout << "Если вы хотите отобразить список - нажмите 6" << endl;
    cout << "Если вы хотите закончить работу со списком - нажмите 0" << endl;
    cout << "Для начала работы программы пожалуйста нажмите нужную цифру" << endl;
    cin >> n;
    while (n != 0){
        if (n == 1) {
            cout << "Введите данные студента в формате:" << endl;
            cout << "Имя Фамилия Студенческий билет(int) Баллы(double) Итоговую отметку в русском формате(int) ";
            cout << "Итоговую отметку в английском формате(char)" << endl;
            cin >> name >> surname >> student_ID >> points >> russian_grade >> english_grade;
            add_to_start( & Head, name, surname, student_ID, points, russian_grade, english_grade);
        }
```

Рис 8

```
        if (n == 2) {
            cout << "Введите данные студента в формате:" << endl;
            cout << "Имя Фамилия Студенческий билет(int) Баллы(double) Итоговую отметку в русском формате(int) ";
            cout << "Итоговую отметку в английском формате(char)" << endl;
            cin >> name >> surname >> student_ID >> points >> russian_grade >> english_grade;
            add_to_end( & Head, name, surname, student_ID, points, russian_grade, english_grade);
        }
        if (n == 3) {
            cout << "Введите данные студента в формате:" << endl;
            cout << "Студак того, после кого он должен быть Имя Фамилия Студенческий билет(int) Баллы(double) ";
            cout << "Итоговую отметку в русском формате(int) Итоговую отметку в английском формате(char)" << endl;
            cin >> student_ID_after >> name >> surname >> student_ID >> points >> russian_grade >> english_grade;
            insert_after( & Head, name, surname, student_ID, points, russian_grade, english_grade,  student_ID_After: student_ID_after);
        }
        if (n == 4) {
            cout << "Введите данные студента в формате:" << endl;
            cout << "Студак того, до кого он должен быть Имя Фамилия Студенческий билет(int) Баллы(double) ";
            cout << "Итоговую отметку в русском формате(int) Итоговую отметку в английском формате(char)" << endl;
            cin >> student_ID_before >> name >> surname >> student_ID >> points >> russian_grade >> english_grade;
            insert_before( & Head, name, surname, student_ID, points, russian_grade, english_grade,  student_ID_Before: student_ID_before);
        }
        if (n == 5) {
            cout << "Введите номер студенческого билета студента, которого вы хотите удалить из списка" << endl;
            cin >> student_ID;
            remove_from_list( & Head, student_ID);
        }
```

Рис 9

```
        if (n == 6) {
            print(Head);
        }
        cin >> n;
    }
    return 0;
}
```

Рис 10

Вывод

Выполнение лабораторной работы 1 прошло успешно, все цели выполнены, полностью рабочая программа и отчёт по ней выложены на гитхаб.