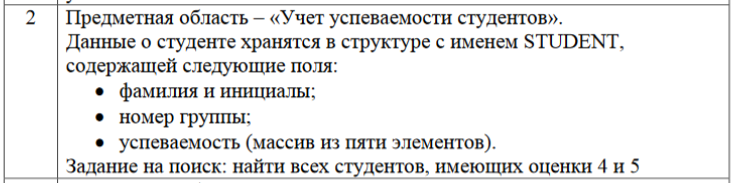
Белов А.И. гр.4932

**Описание предметной области**



Нашей целью выступает написание программы для хранения и анализа данных о студентах. Такое приложение может быть полезным для преподавателей учебных заведений, так как сильно упростит учет успеваемости всех студентов, занимающихся предметом. Поиск студентов с оценками 4 и 5 может пригодиться в том случае, если преподаватель планирует поощрять учеников-хорошистов. В программе будет предусмотрено 5 основных функций, которые вызываются введением в консоль их порядкового номера:

1-ая функция - Вывести всех студентов

2-ая функция – Добавить студента, программа попросит на ввод имя, номер группы, и пять оценок.

3-ая функция – Удалить студента, программа попросит на ввод его порядковый номер в списке.

4-ая функция – Изменить данные о студенте, программа попросит на ввод его порядковый номер в списке, после чего запросит новые имя, номер группы, и оценки.

4-ая функция – Найти студентов по ФИО и вывести информацию о них.

5-ая функция – Отсортировать список студентов по фамилии.

6-ая функция – Отсортировать список студентов по номеру группы.

7-ая функция – Записать все данные в файл.

8-ая функция – Вывести список студентов с оценками 4 и 5.

Чтобы выйти из приложения, необходимо ввести цифру 9.

Все изменения автоматически заносятся в текстовый файл на локальном носителе

Работа реализуется с помощью:

Класса Student, в котором хранятся ФИО, номер группы и массив из 5 оценок, а также функции для работы с одним студентом.

Структуры Node, которая представляет собой узел двухсвязного списка. Хранит объект класса Student, и указатели на предыдущий и следующий узел.

Класса DataStudent, в котором реализуется работа со списком. Хранит в себе указатель на корневой элемент списка и функции для работы.

При сохранении в текстовый файл данные будут записываться в формате name:num:notes[0]:notes[1]:notes[2]:notes[3]:notes[4]. Каждый объект заносится в отдельную строку.

**Псевдокод и блок-схемы**

**Main:**

**Class Stdent:**

|  |  |
| --- | --- |
| **Конструктор Student(**char\* fullname, int group, int notes[n]**)**  Student.fullname = fullname  Student.group=group  Цикл от 0 до n  Student.notes[i] = notes[i] |  |
| **newStudent()**  Ввести name и group  Для каждого элемента массива notes объекта student  Ввести note и вставить в массив |  |

|  |  |
| --- | --- |
| **showStudent()**  Вывести name, num  Для каждого элемента массива notes объекта student  Вывести note |  |

**Struct Node:**

|  |  |
| --- | --- |
| **Конструктор Node()**  Student.newStudent() |  |

|  |  |
| --- | --- |
| **Конструктор Node(Student newstudent)**  Node.student=newstudent |  |

**Class DataStudent:**

|  |  |
| --- | --- |
| Int getLength()  Node linker = root  Int length=0  Пока linker не равен null  linker = linker ->next  length++  Вернуть length |  |

|  |  |
| --- | --- |
| Node\* getNode(int index)  Node linker = root  Пока index не равен 0  linker = linker ->next  index-1  Вернуть linker |  |

|  |  |
| --- | --- |
| Void insertInData()  Если root = null  root = new Node()  root.next=null  root.prev=null  Иначе  Node linker = root  Node newnode = new Node()  Пока linker.next не равен null  linker=linker.next  newnode.prev = linker  linker.next = newnode |  |

|  |  |
| --- | --- |
| Void deleteFromData()  Int index  Ввести index  Если index больше getLength()  Вывести сообщение об ошибке  Иначе если index = 0  Node temp = root  root =root.next  Удалить temp  Иначе  Node linker = root  Пока index не равен 0  linker=linker.next  index-1  Node temp = linker.prev  linker.prev = null  temp.next = linker.next  Удалить linker; |  |

|  |  |
| --- | --- |
| Void changeData()  Int index  Ввести index  Если index больше длины списка  Вывести сообщение об ошибке  Иначе если index = 0  root.newStudent()  Иначе  Node linker = root  Пока index не равен 0  linker=linker.next  index-1  linker.newStudent() |  |

|  |  |
| --- | --- |
| Void findInData()  Char\* findname  Ввести findname  Node linker = root  Пока linker не равен null  Если findname=linker.student.fullname()  linker.student.showStudent()  linker=linker.next |  |

|  |  |
| --- | --- |
| Void showData()  Node linker = root  Пока linker не равен null  linker.student.showStudent()  linker=linker.next |  |

|  |  |
| --- | --- |
| Void sortByGroup()  Int length = getLength()+1  Цикл от 0 до length, счетчик i  Цикл от length-1 до i+1 включительно, счетчик j  Если getNode(j).student.group<getNode(j- 1).student.group  Поменять местами getNode(j) и getNode(j-1) |  |

|  |  |
| --- | --- |
| Void sortByFullName()  Int length = getLength()+!  Цикл от 0 до length, счетчик i  Цикл от length-1 до i+1 включительно, счетчик j  Если getNode(j).student.fullname[0]<getNode(j- 1).student.fullname[0]  Поменять местами getNode(j) и getNode(j-1) |  |

|  |  |
| --- | --- |
| Void readFromFile(char\* pathFile)  Char\* line  Открыть файл in по пути pathfile  Пока in не кончится  Считать строку в line  Int group  Char\* fullname  Int notes[n]  Int i=0  Пока line[i] не равен «:»  fullname[i]=line[i]  i+1  i+1  Пока line[i] не равен «:»  group\*10  group+line[i]  i+1  i+1  Цикл от 0 до 5, счетчик j  Notes[j]=line[i+j]  i+1  Создать объект student(fullname, group, notes)  insertInData(student)  Закрыть in |  |

|  |  |
| --- | --- |
| Void writeToFile(char\* pathFile)  Открыть файл out по пути pathfile  Node linker = root  Цикл от 0 до getLength()+1, счетчик i  Записать в out linker.student.fullname + «:»  Записать в out linker.student.fullname + «:»  Цикл от 0 до n  Записать в out «:»+linker.student.notes(j)  Если i не равен getLength  Переход на след. строку  linker=linker.next  Закрыть out |  |

|  |  |
| --- | --- |
| Void findOtl()  Node linker = root  Пока linker не равен null  Bool otl = true  Цикл от 0 до n, счетчик i  Если linker.student.notes[i]<4  otl=false  Если otl=true  Linker.student.showStudent()  linker=linker.next |  |