Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет прикладной математики и информатики
Кафедра вычислительной математики и программирования

Курсовой проект
По дисциплине
«Практикум на ЭВМ»
2 семестр

Задание VI
«Обработка последовательной файловой структуры на язык Си»

Студент:	Сикорский А. А.
Группа:	М8О-108Б-20
Преподаватель:	Трубченко Н. М.
Подпись:	
Оценка:	
Дата:	20.05.2021

Москва
2021

Содержание

Задание

Разработать последовательную структуру данных для представления простейшей базы данных на файлах в СП Си в соответствии с заданным вариантом. Выяснить, сколько студенток группы p получают стипендию. Параметры задаются с помощью ключей –f (распечатка файла) или –p <parameter> (параметры конкретного варианта задания). Получение параметров из командной строки производится с помощью стандартных библиотечных функций argc и argv.

1 Программа

1.1 Структура данных

Текстовый файл считывается и сохраняется в линейный двунаправленный список. Это такая структура данных, написанная на указателях, в которой нам гарантируется вставка и удаление за $O(1)$. Еще к преимуществам можно отнести потенциально больший размер списка по сравнению с массивом, ведь список не требует последовательного выделения памяти, но за это стоит платить невозможностью прямого доступа к элементу, как мы это делаем в массиве по индексу. Двунаправленность значит, что в каждом элементе списка содержится указатель не только на следующий, но и на предыдущий элемент. Таким образом доступна навигация как влево, так и вправо. Сама навигация по списку и его функции работают с помощью итераторов. Так нам не особо важна реализация списка. Используя итераторы, мы можем поменять внутреннее устройство списка и изменить только реализацию итераторов. При этом остальные функции по типу удаления, поиска и сортировки не сломаются. У списка есть следующие функции:

1. **Create** выделяет память под *head* и инициализирует пустой список.
2. **Insert** вставляет элемент перед переданным как аргумент итератором.
3. **Delete** удаляет элемент на месте, куда указывает итератор.
4. **Size** возвращает размер списка.
5. **isEmpty** проверяет список на пустоту.
6. **Write** записывает данные по итератору.
7. **First и Last** возвращают итераторы на первый и терминирующий элементы соответственно.

8. **Read** считывает элемент по итератору.

9. **nPos** возвращает итератор, указывающий на n элемент списка.

10. **Next и Prev** двигают переданный итератор вперед и назад по списку.

11. **Equals** проверяет равенство двух итераторов.

12. **move** двигает итератор на n элементов вправо.

13. **Destroy** очищает список.

1.2 Работа с аргументами и выполнение задания

Для нормальной работы программе нужны 2 аргумента в случае, если нужно распечатать файл, и 3 аргумента, когда нужно выполнить задание. Программа работает на примере ПМИ МАИ 1 курс. Т.е имеем доступные к рассмотрению группы 106, 107, 108. Программа дает пользователю понять, как и какие аргументы нужно вводить при запуске. То есть проверяются на правильность сами ключи и параметр, указывающий на обрабатываемую группу. Если что-то введено неверно, то выполнение программы завершается с ошибкой, а пользователь получает сообщение-справку о том, что ему нужно вводить.

Для выполнения задания варинта заводим счетчик людей, получающих стипендию, и проходим по списку, проверяя людей на соответствие условиям. Получается, что изначально мы считаем каждого просматриваемого человека за стипендиата. Если он сохраняет это звание к концу проверки (у него не обнаруживаются тройки или двойки), то инкрементируем счетчик. Условиями являются соответствующий заданному номер группы, женский пол и все оценки 4 - 5. Если же аргументом подается -f, то программа просто распечатывает файл.

Листинг 1: Задание варианта

```
    int counter(iterator* first, iterator* last, unsigned number) {
    int badBoys = 0;
    while (!Equals(first, last)) {
        bool isStipuha = true;
        Person p = Read(first);
        if (!(p.mark1 > 3 && p.mark2 > 3 && p.mark3 > 3 && p.mark4 > 3))
            isStipuha = false;
        if (isStipuha && p.group == number && !strcmp(p.sex, "f"))
            badBoys++;
        Next(first);
    }
    return badBoys;
}
```

В функцию передаются итераторы на конец и начала списка а также переменная-номер группы, принятой к рассмотрению. Итератор *first* является итератором, по которому мы и читаем какой-то элемент в списке. Движение по списку продолжается, пока данный итератор не добрался до итератора на терминирующий элемент *last*.

2 Вывод

В ходе работы я использовал линейный двусвяный список для обработки файла и выполнения задания по подсчету людей, получающих стипендию в выбранной группе. Оценим сложность обработки, которая составляет $O(n)$ так как элементы просматриваются последовательно друг за другом в цикле.