ЛАБОРАТОРНАЯ РАБОТА №6

Цель работы:

Приобрести умения и практические навыки для работы по организации стандартного файлового ввода-вывода и организации произвольного доступа к файлам; приобрести умения работы с линейными списками.

Теоретическая часть:

Существует два способа представления файлов:

– *text* – текстовое – доступно распознавание и обработка конца строки;

– *binary* – двоичное – доступен каждый байт.

Файл становится доступе после открытия.

FILE – предопределенная структура, содержащая информацию для работы с файлами.

Синтаксис открытия файла:

```
errno_t fopen_s (FILE *f, const char * fname, const char * fmode);
```

Строка *fname* содержит имя или адрес имени файла. Строка *fmode* определяет режим открытия файла.

Существуют несколько режимов открытия файла:

– *w* – открыть файл для записи (если файл не существует, он будет создан, если существует – усечен до нуля);

– *w+* – открыть файл для обновления (для чтения и записи) (если файл не существует, он будет создан, если существует – усечен до нуля);

– *r* – открыть файл для чтения;

– *r+* – открыть файл для обновления (для чтения и записи), при этом запись производится в любое место;

– *a* – открыть файл для записи, данные записываются в конец файла, если файл не существует, то он создается;

– *a+* – открыть файл для обновления (для чтения и записи), запись производится в конец файла, если файл не существует, то он будет создан, читать можно весь файл.

Функция fopen_s создает указатель на структуру, содержащую

информацию о файле, если файл успешно открыт. Функция fopen_s возвращает 0, если файл успешно открыт, или ненулевое значение в противном случае.

Операции ввода-вывода выполняются с текущей позиции файла, определенной указателем потока, который изменяется автоматически после каждого ввода-вывода. При открытии файла он ставится в начало (*w*, *r*) или в конец (*a*).

Закрытие файла производится при помощи следующей функции:

```
int fclose (FILE * f);
// 0, если закрыт, EOF в противном случае
```

Также файл закрывается при завершении программы.

EoF – специальное значение, возвращаемое в программу некоторыми файлами. Оно определяется по-разному, но не соответствует ни одному символу.

Перед закрытием файла информация из связанных с ним буферов выгружается на диск. Во избежание потери информации рекомендуется закрывать файл явно.

Ввод-вывод информации в файл производится при помощи следующих функций:

– *int fscanf_s (FILE * f, const char * FMT, ...);*
– *int fprintf (FILE * f, const char * FMT, ...);*

Конец файла – попытка чтения за пределами файла, ведет к остановке. Определение конца файла:

```
int feof (FILE * f);
```

Избежать проблемы чтения из пустого файла поможет цикл с предусловием, при этом чтение из файла необходимо производить перед циклом.

```
#include "stdafx.h"
FILE * f;
int n, i;
void main(){
if (fopen_s(&f,"a1","r"))
     printf("\n ошибка открытия");
else {
     fscanf_s(f,"%d",&n);
```

```
        while(!feof(f)){
            printf("%d\n",n);
            fscanf_s(f,"%d",&n);
        }
        fclose(f);
        }
}
```

Обработка ошибок:

```
int ferrot (FILE * f);
```

Если возвращаемое значение не 0, то есть какая-то ошибка, иначе ошибок нет.

Двоичный ввод-вывод работает с информацией в двоичном виде.

```
fread (void * buf, size_t size, size_t const, FILE * f);
```

Функция считывает *const* элементов размера *size* в область указателя *buf* из потока *f*. Если количество элементов совпадает, то все хорошо, если меньше, то возникает ошибка или *EoF*.

Вывод из *buf* числа *const* элементов размера *size* в поток *f*:

```
fwrite (void * buf, size_t size, size_t const, FILE * f);
```

Работа с файлом – установка, чтение указателя в файле – производится при помощи специальных функций.

```
long int ftell (FILE * f);
```

Данная функция возвращает текущую позицию указателя.

```
int fseek (FILE * f, long off, int org);
```

Данная функция осуществляет перемещение на позицию *off*, отсчитанную от значения *org*, которое может быть равно 0 (*SEEK_SET*), 1 (*SEEK_CUR*), 2 (*SEEK_END*).

```
int fgetpos (FILE * f, fpos_t *pos);
```

Эта функция возвращает текущую позицию указателя и копирует его по адресу *pos*.

```
int setpos (FILE * f, fpos_t *pos);
```

Данная функция осуществляет перемещение текущей позиции на *pos* из предыдущей функции (отсчет производится от начала файла).

При изменении направления движения информации нужно явно устанавливать позицию указателя в файле.

Основные структуры данных – массивы, записи, множества. Из них формируются более сложные структуры. Структуры данных – сами элементы и связи между ними. Каждая структура характеризуется набором операций: включение, поиск, удаление. В основе структур данных лежат списки.

Список – упорядоченный набор элементов, в котором выделен первый элемент и каждый имеет одного предшественника (кроме последнего в списке).

Существует два способа представления списков в памяти:

– *последовательное* – элементы физически следуют в памяти друг за другом;

– *связанное (ссылочное)* – каждый элемент содержит ссылку на следующий – потенциально бесконечный или кольцевой список.

Линейный список – конечная упорядоченная совокупность элементов. *Упорядоченность* – заранее определенный порядок доступа к элементам, соответствующий логическому порядку их следования в списке. Доступ к ним осуществляется при помощи указателя списка – первого элемента списка.

С линейными списками можно производит определенные действия – включение элемента, удаление элемента, проход по списку.

При ссылочном представлении память под отдельные компоненты структуры выделяют в процессе выполнения программы, а на этапе трансляции память выделяется для хранения адреса динамически размещаемой переменной. При этом отдельные компоненты структуры могут размещаться в несмежных областях памяти и связываются между собой при помощи адресов или ссылок. Поэтому каждый элемент списка помимо содержательной информации имеет ссылки на следующий элемент. Таким образом, каждый элемент списка – структура, состоящая как минимум из двух полей (информация и указатель).

```
struct Els {
     int data;
     Els * next;
} * p;
```

Здесь *data* – информативное поле, *next* – ссылочное поле, а *p* – адрес начала списка.

Если мы обращаемся к полям структуры через указатели используется следующий синтаксис:

```
имя->поле
```

Действия над списком:

– добавление в начало (если p = 0 – адрес текущего элемента списка, то будет создан новый список);

```
Els *q = (Els*) malloc (sizeof (Els));
q->next = p;
p = q;
```

– добавление после p-го элемента;

```
Els *q = (Els*) malloc (sizeof (Els));
q->next = p->next;
p->next = q;
```

– добавление перед p-ым элементом;

```
Els *q = (Els*) malloc (sizeof (Els));
*q = *p;
p->data = <новое значение>;
p->next = q;
```

– удаление за p-ым элементом;

```
Els *q = p->next;
p->next = q->next;
free (q);
```

– удаление элемента p;

```
Els *q = p->next;
*p = *q;
free (q);
```

– проход по списку.

```
while (!p) {
//обработка элемента
p = p->next;
}
```

Задание 1.

Описать структуру T, содержащую следующие сведения:

1. Фамилия автора, название книги, издательство, год издания.
2. Название автомобиля, марка, страна производитель, год выпуска.
3. Фамилия человека, имя человека, возраст, пол.

Написать функции, выполняющие следующие действия:

- Создание двоичного файла FB.txt, содержащего N элементов типа T. Функция возвращает 0 при успешном завершении и -1 в противном случае. Параметр N и значения вводятся с клавиатуры.
- Вывод на экран содержимого файла FB.txt. Функция возвращает в вызывающую программу количество записей через параметр K, а также 0 при успешном завершении и -1 в противном случае.
- Возврат в вызывающую программу следующего значения:

1.1. Количество книг указанного автора.

1.2. Количество книг указанного года издания.

1.3. Количество книг указанного издательства.

2.1. Количество автомобилей указанной марки.

2.2. Количество автомобилей из указанной страны.

2.3. Количество автомобилей указанного года выпуска.

3.1. Количество людей с указанной фамилией.

3.2. Количество людей указанного пола.

3.3. Количество людей указанного возраста.

- Возвращение в вызывающую программу записи под номером n.
- Изменение значения следующего поля для n-го элемента файла:

1.1. Фамилия автора.

1.2. Год издания.

1.3. Издательство.

2.1. Марка автомобиля.

2.2. Название страны производителя

2.3. Год выпуска автомобиля

3.1. Фамиля человека.

3.2. Пол человека.

3.3. Возраст человека.

Написать программу, содержащую меню из заданных функций. Программа завершает работу при выборе пункта меню «Выход»

Задание 2.

Создать список из 10 элементов. Каждый элемент содержит строку с указанием на номер элемента вида «Это … элемент». Длина каждой строки не более 20 символов. Составить функции для:

- Печати списка
- Удаления из списка

1. первого элемента,
2. второго элемента,
3. первых двух элементов,
4. k-го по порядку элемента,
5. всех элементов, начиная с k -го,
6. предпоследнего элемента,
7. последнего элемента,
8. последних двух элементов,

- Добавления в список нового элемента

1. после первого элемента,
2. перед первым элементом,
3. после второго элемента,
4. перед вторым элементом,
5. после k-го по порядку элемента,
6. перед k-ым по порядку элементом,
7. после последнего элемента,
8. перед последним элементом,

Вопросы к теоретическому материалу

1. Какая функция используется для открытия файла?
2. Какие существуют режимы открытия файла?
3. Какая функция используется для закрытия файла?
4. Какая функция применяется для чтения информации из файла?
5. Какая функция применяется для записи информации в файл?
6. Какая функция возвращает текущую позицию указателя?
7. Какая функция осуществляет перемещение указателя на указанную позицию?
8. Что называется списком?
9. Какие способы представления списков в памяти существуют?
10. Охарактеризуйте последовательное представление списков в памяти.
11. Охарактеризуйте ссылочное представление списков в памяти.
12. Что называется линейным списком?
13. Какие действия можно производить с линейными списками?
14. Укажите типовой синтаксис элемента списка.
15. Опишите, каким образом добавляется элемент в начало списка.
16. Опишите, каким образом добавляется элемент после *p*-го.
17. Опишите, каким образом происходит удаление элемента после *p*-го.
18. Опишите, каким образом происходит удаление элемента *p*;
19. Опишите, каким образом происходит проход по списку.

ПРОЦЕСС СДАЧИ ЛАБОРАТОРНОЙ РАБОТЫ

По итогам выполнения каждой лабораторной работы студент:

1. Демонстрирует преподавателю правильно работающие программы;
2. Демонстрирует приобретенные теоретические знания, отвечая на пять вопросов по лабораторной работе;
3. Демонстрирует отчет по выполненной лабораторной работе, соответствующий всем требованиям.

Отчет по лабораторной работе оформляется по шаблону, представленному в приложении 1. Требования к отчету представлены в приложении 2.

ПРИЛОЖЕНИЕ 1. ШАБЛОН ОТЧЕТА

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Казанский национальный исследовательский технический университет
им. А.Н. Туполева – КАИ»
Институт компьютерных технологий и защиты информации
Отделение СПО ИКТЗИ (Колледж информационных технологий)

ЛАБОРАТОРНАЯ РАБОТА №
по дисциплине
СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ

Работу выполнил
Студент гр.43__
Фамилия И.О.
Принял
Преподаватель Григорьева В.В.

Казань, 2025 г.

1. **Цель работы.**

2. **Задание на лабораторную работу** – вставляется задание на лабораторную работу, соответствующее индивидуальному, выданному преподавателем, варианту студента.

3. **Результат выполнения работы** – формируется описание хода выполнения работы и вставляются скриншоты с результатами работы разработанных программ (скриншоты должны быть подписаны, например, *Рисунок 1. Начальное состояние программы* и т.п.).

4. **Листинг программы** – вставляется код разработанной программы

ПРИЛОЖЕНИЕ 2. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Лист документа должен иметь книжную ориентацию, поля документа должны составлять: левое – 3 см, правое – 1,5 см, верхнее – 2 см, нижнее 2 см.

Нумерация страниц – внизу страницы по центру, первая страница не нумеруется

Междустрочный интервал – 1,5 (полуторный), отступ первой строки – 1,25.

Текст документа должен быть выполнен с использованием шрифта Times New Roman, размер – 14, выравнивание – по ширине. Заголовки выполняются тем же шрифтом, но размера 16, полужирное начертание, размещение – по центру.

Рисунки должны размещаться по центру, они нумеруются по порядку. Перед рисунком в тексте на него должна быть ссылка. Подпись рисунка должна располагаться по центру и быть выполнена шрифтом Times New Roman, размер – 12. Сначала происходит нумерация рисунка, а затем пишется его название.