1 ВВЕДЕНИЕ

1.1 Цель проведения вычислительной практики

Основной целью учебной практики студентов направлений 230100, 09.03.01 является закрепление теоретических знаний, умений и практических навыков структурного программирования с использованием основных алгоритмов нечисловой обработки данных. В соответствии с “Рабочей программой” вычислительная практика включает 3 задания по следующим темам:

1) Освоение методов работы с бинарными и текстовыми файлами, освоение алгоритмов удаления данных из бинарного файла.
2) Освоение алгоритмов сортировки структурированных данных бинарных файлов, с использованием динамических объектов - массивов, списков.
3) Приобретение навыков разработки многомодульных программ, включающих библиотеку пользовательских функций, средства условной компиляции и систему меню для выбора вариантов обработки данных.

1.2 Порядок выполнения работ вычислительной практики

1) Студент получает от преподавателя варианты каждого практического задания и консультацию по разработке алгоритмов и текстов программ.
2) Самостоятельно разработав схему алгоритмов очередного практического задания, студент получает допуск у преподавателя к выполнению его.
3) Выполнение задания проводится во время занятий в классах ЦИТСО МГТУ ГА в присутствии преподавателя.
4) После выполнения очередного задания результаты выводятся на печать, и оформляется соответствующий раздел отчета по вычислительной практике.
5) Производится поэтапная защита результатов вычислительной практики по мере выполнения очередного задания и оформления соответствующего раздела отчета. Успешная защита каждого задания фиксируется отметкой в журнале.
6) На зачете по вычислительной практике студент представляет отчет, оформленный в соответствии с Единой системы программной документации (ЕСПД) и государственным стандартом (ГОСТ 19.701- 90), включающий отчеты по каждому заданию, отвечает на любые вопросов, связанные с выполнением практических заданий и получает результирующую отметку.

1.3 Оформление отчета по вычислительной практике

Отчет по результатам прохождения вычислительной практики должен включать отчеты по каждому практическому заданию. Отчет содержит текстовую часть, рисунки, схемы алгоритмов, которые должны быть выполнены в соответствие с ГОСТом.

Отчет должен иметь следующую структуру:

1) титульный лист;
2) аннотация;
3) содержание;
4) содержательная часть - разделы по каждому практическому заданию;
5) приложения, которые содержат тексты программных файлов, экранные формы и т. п.

В **аннотации** кратко, тремя – четырьмя предложениями раскрывается цель и основные результаты, полученные при выполнении вычислительной практики. Аннотация выполняется на отдельном листе.

Содержание отчета включает номера и наименования разделов и подразделов с указанием номеров страниц. Пример оформления содержания дан на рис. 1.1.

Содержание

Рис. 1.1 Пример оформления содержания

Рекомендуемая форма **титульного листа** представлена на рис. 1.2.

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ»

Кафедра вычислительных машин, комплексов, систем и сетей

Отчет защищен с оценкой

_____

_____
(подпись преподавателя, дата)

ОТЧЕТ
по результатам выполнения Вычислительной (учебной) практики
по дисциплине ”Программирование”

Выполнил студент группы ЭВМ 1-1

_____
(Ф.И.О., № зачетной книжки)

_____
(подпись, дата)

МОСКВА - 2014

Рис. 1.2 Пример титульного листа

2 Практическое задание №1

2.1 Цель

Освоение методов работы с бинарными и текстовыми файлами, освоение алгоритмов удаления данных из бинарного файла.

2.2 Техническое задание

Разработать схемы алгоритмов, структуру и текст программы для обработки структурных данных текстового и бинарного файлов с использованием функций создания файлов и удаления ряда структур из файлов.

Схемы алгоритмов должны быть разработаны для каждой функции.

Тестирование функции удаления проводить с помощью файла данных для удаления, при этом должны быть либо удалены отмеченные записи из бинарного файла и текстового файла, либо выведены правильные фразы, например об отсутствии искомых данных.

2.3 Порядок выполнения

1) Объявить структурный тип, для описания характеристик объекта вашего варианта. Объявить внешнюю структуру.

Например:

struct stud { char fio [20]; char gr [7]; unsigned int no ; float rs; } st ;

2) Объявить входной, выходной и двунаправленный файловые потоки.

3) Определить функцию (***filtr()***) для удаления пробелов (ведущих и замыкающих) в строковых данных.

4) Объявить функцию перегрузки операций вставки в поток "<<" (операции вывода) и извлечения из потока ">>" (операции ввода) для данного структурного типа по формату текстового файла данных, который будет программно создаваться.

При перегрузке операции вывода "<< " на каждое поле структуры выделять нужное количество позиций и производить вывод с левым выравниванием, вывод полей вещественных типов производить с двумя десятичными знаками.

Операция ввода данных из потока должна учитывать форматы файла данных.

Ниже приведены функции перегрузки операций ввода / вывода для структурного типа ***stud***:

```
// перегрузка операции  вывода структурных данных в стандартный
//выходной поток и во все  выходные потоки производных типов (напр.,
//файловые)
#include <iostream>
#include <iomanip>
using namespace std;
…
ostream &     operator <<    ( ostream & out , stud &st) {
out.setf(ios::left,ios::adjustfield);    //сбросили правое выравнивание
                                         //установили  левое
out << setw(20) << st.fio << setw(7) <<
st.gr << setw(8) <<st.no   << setw(7)<< setprecision(2) << st.rs<<endl;
return out;
 }
//---------------------------------------------------------------------------
// перегрузка операции ввода данных (“>>”) из стандартного потока,
// наследуемая и файловыми входными потоки
istream &   operator >>   (istream & in , stud &st ) {
char T[80];
in.getline (st.fio, 21);  filtr( st.fio);
in.getline ( st.gr, 8) ;  filtr(st.gr );
in >> st.no >>st.rs ;
in.getline (T, 80);
return in;
 }
```

5) Определить функцию создания текстового файла данных **sozdat().** Каждая строка файла будет содержать элементы одной из структур, вводимых с клавиатуры.

Параметр функции - имя создаваемого текстового файла. В теле функции создается файл с данным именем для записи данных в текстовом режиме.

Далее алгоритм функции строится в диалоговом режиме:

Функция запрашивает данные. С клавиатуры вводятся значения полей и заносятся в оперативную память внешней структуры **st**. Значения введенных строковых полей “очищаются” от ведущих и завершающих пробелов. Ввод осуществляется с помощью перегруженной операции ввода данных из стандартного потока “>>”. Затем структура “целиком” из оперативной памяти копируется в текстовой файл, используя перегруженную операцию вывода “<<”.

Далее следует запрос о продолжении записи в файл. В случае положительного ответа, продолжается ввод/вывод данных, в противном случае,

функция завершает свою работу с помощью оператора **return** , при этом перед завершением закрывается созданный файл.

```
void sozdat (char*file) {
fout.open(file) ;                // fout- файловый выходной поток
m: cout<< "введите данные одной структуры"<<endl ;
cin>> st;    fout<<st;
cout<<"Нажмите любую клавишу, если хотите продолжить запись и  "
            " 0 – , если нет" <<endl
char c ;    c=getch();
 if(c=='0') {fout.close(); return;}
else  goto m;
 }
```

6) Определить функцию создания бинарного файла.

Бинарный файл содержит плотно упакованные двоичные представления структур, при этом составляющие их элементы располагаются подряд без разделения и различные структуры также друг от друга ничем не разделяются, так, например, нет деления на строки, содержащие данные одной структуры как в текстовом файле.

Параметрами функции создания являются имена файлов: текстового файла с данными и создаваемого бинарного файла.

В теле функции открывается файл данных для чтения данных в текстовом режиме.

Новый файл создается для записи данных в файл в бинарном режиме.

Организуется цикл, в котором, пока не будет достигнут конец файла данных, из файла считываются построчно данные в переменную **st,** используя перегруженную операцию извлечения (">>") и "целиком" **st** записывается в бинарный файл, используя компонентную функцию выходных потоков **write()** для двоичного (бинарного) вывода данных.

Функция завершается закрытием файлов.

7) Создать с клавиатуры файл для тестирования функции удаления данных из бинарного файла. В каждую строку этого файла записать значение одного из полей структур, по которому будет производиться поиск удаляемых структур в бинарном файле.

Файл следует составить следующим образом:

- значение, равное полю последней структуры бинарного файла;
- отсутствие поискового признака (пустая строка);
- значение, не совпадающее с полями структур в бинарном файле;
- значение, совпадающее с полем первой структуры бинарного файла.

8) Определить функцию удаления записей из бинарного файла. Параметрами функции являются имена файлов: бинарного файла, в котором

хранятся структурные данные и файла данных – с поисковыми данными для удаления.

Открыть файлы:

- файл данных для удаления в режиме текстового чтения;
- файл с записями структурных данных (бинарный) в режиме бинарного чтения и записи;
- новый рабочий файл для записи данных в бинарном режиме.

Объявить переменную, в которую будут читаться данные для удаления.

Организовать цикл (пока не будет достигнут конец файла данных для удаления), в теле которого выполнять следующие действия:

- считывать значение для удаления;
- если значение для удаления представляет собой символьную строку, удалить из этой строки пробелы, ведущие и замыкающие;
- если значение для удаления отсутствуют (например, это пустая строка), вывести на экран (и в файл результатов) соответствующую фразу об отсутствии поискового признака и с помощью оператора **continue** перейти к следующей итерации цикла;
- в противном случае, если данное для удаления не пустое значение, выполнить следующие действия:
    1. установить указатель в бинарном файле с записями на начало файла;
    2. организовать вложенный цикл (до конца бинарного файла), в теле которого:
        - считывать по одной структуре из файла в переменную **st,** используя компонентную функцию входных потоков **read()** для двоичного (бинарного) ввода данных;
        - проверять совпадение контрольного поля структуры **st** со значением для удаления;
        - в случае совпадения, надо присвоить этому полю структуры **st** характерное значение, например, пустую строку (“пометить” удаляемую структуру);
        - и перезаписать откорректированную структуру на старое место в бинарный файл (для этого надо переместить указатель записи в файле “назад” на одну структуру);
    3. если по завершению внутреннего цикла были прочитаны все записи бинарного файла, и совпадение полей не обнаружилось, вывести фразу, что таких данных в бинарном файле нет;

Когда завершится внешний цикл, следует опять установить указатель в бинарном файле на начало.

Организовать цикл, в котором считывать по одной записи из бинарного файла в структуру **st**, используя функцию **read()** двоичного чтения. Если значение контрольного поля в структуре **st** отличается от характерного

значения (пустой строки), следует записывать структуру в рабочий файл, используя функцию **write()** . Цикл выполнять до конца бинарного файла.

Закрыть бинарный и рабочий файлы. Бинарный файл удалить с диска, а рабочему файлу дать имя бинарного файла.

Вызвать функцию коррекции файла данных.

9) Функция коррекции файла данных.

Параметры функции имена файлов: текстового файла данных и бинарного файла содержащего записи структур.

Открыть бинарный файл в режиме бинарного чтения.

Открыть файл данных для записи данных в текстовом режиме (создается новый файл)

Организовать цикл (до конца бинарного файла), в котором считывать из бинарного файла по одной записи в оперативную память структуру **st** и выводить эту структуру “целиком” в файл данных, используя перегруженную операцию **"<<".**

Закрыть файлы.

10) Функция чтения бинарного файла**.**

Алгоритм и текст функции чтения записей бинарного файла были разработаны на лабораторной работе №6. Следует воспользоваться ими.

Примечание:
1) проводить контроль открытия файлов;
2) проводить контроль переименования и удаления файлов;
3) перед определением каждой функции должен быть комментарий о назначении функции.

2.4 Алгоритмы удаления данных из файла:

1) Алгоритм аналогичен алгоритму, представленному выше. Отличие состоит в том, что после того как рабочий файл сформирован, следует переписать записи из рабочего файла в основной бинарный файл;

2)Если в бинарном файле найдена структура, которую следует удалить, то на ее место в файле копируется структура, следующая за ней, на которую, в свою очередь также копируется следующая структура и так далее до конца файла. В результате такого копирования удаляемая структура будет удалена (а именно - "закрыта" следующей за ней структурой), но в конце файла окажутся две одинаковые последние структуры, поэтому необходимо усечь файл на размер одной структуры. Реализация данного алгоритма должна базироваться

на использовании функций ввода / вывода нижнего уровня (например, использовать **chsize**() – функцию изменения размеров файла).

3)"Пометить" удаляемые структуры в бинарном файле каким-то характерным значением контрольного поля (например, пустой строкой).

Создать связанную динамическую структуру – очередь, из "непомеченных" структур файла. Перезаписать данные связанной структуры в бинарный файл.

2.5 Вопросы к защите:

1) Поясните алгоритмы функций и программу.
2) Поясните разницу между текстовым и бинарным файлом.
3) Что такое связанные структуры? В чем состоит разница между стеком, очередью и списком?
4) Поясните механизм перегрузки стандартных операций.
5) Какие еще способы удаления записей из бинарного файла, кроме выполненного варианта вы знаете?

3 Практическое задание №2

3.1 Цель

1) Освоение методов внутренней сортировки.
2) Освоение алгоритмов сортировки структурированных данных бинарных файлов, с использованием динамических объектов - массивов, списков.

3.2 Техническое задание

Разработать схему алгоритма, структуру и текст программы для обработки данных бинарного файла с использованием функции сортировки структурированных данных файла заданным методом.

3.3 Порядок выполнения:

Метод 1

1) Разработать алгоритм и текст функции внутренней сортировки массива структурированных данных заданным методом. Алгоритмы методов сортировки числовых массивов были рассмотрены на лекциях, а также приведены, например, в Справочном пособии по теме “Нечисловая обработка

данных” (1225), автор – Л.М. Климова. Отличие сортировки массива структур состоит в том, что для анализа сортированности сравниваются поля структур (ключи сортировки), а при обнаружении неупорядоченности местами в массиве надо обменивать структуры целиком.

Параметрами функции сортировки массива структур являются массив структур, предназначенный для сортировки, определенный с открытой границей и количество структур в массиве.

2) Разработать алгоритм и определить функцию упорядочения бинарного файла, хранящего записи структур. Параметр функции – имя бинарного файла. В функции определить количество записей структур в бинарном файле. Выделить динамически память под массив структур, заполнить его структурами, считанными из бинарного файла, отсортировать массив, используя функцию сортировки. Создать новый бинарный файл с тем же именем и вывести в него отсортированный массив.

Варианты по методам сортировки массивов:

№ варианта	метод сортировки
1	метод простого выбора
2	сортировка пузырьковым включением
3	сортировка методом Шелла
4	сортировка пузырьковым всплытием
5	быстрая сортировка
6	сортировка последовательным слиянием
7	сортировка распределением
8	сортировка методом подсчета

Метод 2

Работа с линейной связанной структурой - списком, в котором структуры, соединены в порядке убывания или возрастания ключевого признака – одного из элементов структур, составляющих список.

1) Объявить структурный тип для определения связанных структур, объединяющий конкретные данные варианта с указателем для связи структур в список.
2) Определить внешний указатель на вершину списка.
3) Определить функции дополнения списка. Алгоритм и текст функции дополнения списка представлены в лекции «Связанные структуры. Список».

4) Определить функцию вывода структур списка в бинарный и текстовой файлы. Алгоритм функции вывода идентичен алгоритму функции чтения списка, также рассмотренному в лекции. Отличие заключается в том, что вывод структур списка надо производить не на экран, а в файлы бинарный и текстовой.

5) Определить функцию упорядочения бинарного файла. Параметр функции – имя бинарного файла. Заполнить список структурами, считанными из файла, используя функцию дополнения списка. Вывести структуры списка в файлы - бинарный и текстовой, используя функцию вывода списка.

Примечание:
В качестве составной части практического задания №2 использовать нужные методы и данные практического задания №1.

3.4 Вопросы к защите:

1) Что такое сортировка?
2) Что такое ключ сортировки?
3) Внутренняя и внешняя сортировка.
4) Устойчивость метода и его естественное поведение.
5) Методы внутренней сортировки.
6) Суть метода сортировки Вашего варианта
7) Поясните алгоритмы и текст программы.
8) Поясните алгоритм использования списка для упорядочения файла.

4 Практическое задание №3

4.1 Цель

1) Приобретение навыков разработки многомодульных программ.
2) Освоение методов разработки программы, управление которыми, осуществляется с помощью меню.
3) Освоение препроцессорных средств управления ходом компиляции программы.

4.2 Техническое задание

Разработать многомодульную программу, включающую:

1) заголовочный файл, с объявлением внешних переменных и прототипов используемых пользовательских функций;
2) библиотеку функций пользователя – файл с определениями функций;

3) файл с главной функцией, содержащий систему меню для выбора вариантов обработки данных.

Обрабатывать структурные данные, записи которых содержатся в бинарном и текстовом файлах.

Разработать структуру программы и схемы алгоритмов каждой функции, включая главную.

4.3 Порядок выполнения

1)Составить заголовочный файл **lab.h** с определениями типов, глобальных переменных, глобальных констант всей программы, здесь же определить массив указателей типа **char***, инициированный строками шапки таблицы. В файл включить также описания (прототипы) функций, которые будут определены в библиотеке пользователя, и заголовочные файлы стандартных библиотек.

Чтобы предотвратить повторную компиляцию в программе определений тех же объектов использовать блок условной компиляции:

```
// заголовочный файл lab.h
#ifndef  LabH
#define  LabH
//список определений и описаний
. . .
#endif
```

В этом случае все определения будут откомпилированы только при первом подключении файла **lab.h** .

2) Составить файл с определениями всех пользовательских функций программы.

В файле обязательно должны присутствовать определения следующих функций: вывода шапки таблицы, вывода одной строки таблицы, ”фильтрации” строк, перегрузки операций ввода-вывода для данного структурного типа, создания текстового файла данных и бинарного файла, чтения бинарного файла, удаления данных из бинарного файла, упорядочения бинарного файла и коррекции файла данных.

3) Составить третий файл с главной функцией программы. Тексты двух остальных файлов включить в текст этого файла с помощью препроцессорных директив.

Разработать структуру меню для вызова функций каждого вида обработки данных.

4) Тестировать программу.

4.4 Пример файла программы, содержащего меню

```
// заголовочный файл  lab .h
#ifndef  LabH
#define  LabH
#include < fstream.h>
#include < stdlib.h>
#include < stdio.h>
#include < string.h>
#include < conio.h>
ofstream fout , out;
ifstream fin;
 . . .
#endif

// файл с главной функцией lab.cpp
#include “lab.h”
#include “func.cpp” //файл с определениями функций
 void main()
{  // clrscr() ;
out. open ( "lab. res");
if (!out)
{ cout<< "ошибка открытия файла результатов"; exit(0); }
 cout<<" Нажмите любую клавишу, если вы хотите создать или"
    " обновить файл данных "
   "и нажмите 0 - если нет"   << endl ;
 char c=getch() ;
 if(c!=' 0') { cout<<"Создание файла данных"<<endl; sozdat( "lab.dat"); }
 fin.open( "lab.dat");
 if (!fin) { cout<< "Ошибка открытия файла данных"; exit(0); }
  // textbackground (CYAN);
//   textcolor (BLACK);
   do{
     clrscr ();

   cout<<"\n              МЕНЮ : \n"
        << "\n       Создание файла------------------------------ 1"
        << "\n       Чтение файла-------------------------------2"
        << "\n       Сортировка файла--------------------------  3"
        << "\n       Удаление данных из файла----------------  4"
        << "\n       Выход---------------------------------------- Esc";

   cout<<"\n\n Для выполнения операции введите нажмите нужную"
```

```
   " клавишу \n";
  c=getch();
  switch(c)
   { case '1': clrscr(); sozd(); getch();  break;
     case '2': clrscr(); cht(); getch(); break;
     case '3': clrscr();sort(); getch();  break;
     case '4': clrscr();ud(); kordat();  getch(); break;
     case 27 : clrscr(); exit(0); break;
     }
     }   while(c!=27);
     out. close();
 }
```

4.5 Алгоритмы организации меню:

Алгоритм организации меню с помощью массива указателей на функции. В алгоритме отсутствует оператор выбора варианта.

4.6 Вопросы к защите

1) Для чего используется условная компиляция?
2) Поясните назначение всех директив препроцессора, используемых в программе.
3) Методы разработки многомодульной программы.
4) Пояснить назначение и алгоритм “меню”.
5) Пояснить алгоритм организации меню с помощью массива указателей на функции.

5. СПИСОК ЛИТЕРАТУРЫ

1) Подбельский В.В. Стандартный Си++. - М.: Финансы и статистика , 2008.
2) Павловская Т. А., Щупак Ю. А. С++. Объектно-ориентированное программирование. Практикум. – СПб.: Питер, 2006.
3) Павловская Т. А. С/С+. Программирование на языке высокого уровня. – СПб.: Питер, 2003.

СОДЕРЖАНИЕ