ВВЕДЕНИЕ

Лабораторные работы выполняются в среде Microsoft Visual Studio как консольные приложения по следующим темам:

1. Разработка программ с использованием рекурсивных и итерационных функций, использующих параметр-указатель на функцию;
2. Разработка программ с использованием структурированных данных;
3. Программирование с использованием текстовых и бинарных файлов, содержащих структурированные данные.

Каждая работа выполняется в соответствии с конкретным вариантом задания.

По каждой лабораторной работе оформляется отчет, который должен содержать:

- цель;
- вариант задания;
- структуру программы;
- схемы алгоритмов всех функций программы;
- таблицы глобальных переменных программы и локальных переменных каждой функции;
- листинги файлов программы, исходных данных и результатов.

После выполнения лабораторной работы студент должен защитить ее, пояснив процесс обработки данных, схемы алгоритмов и текст программы, а также ответить на ряд контрольных вопросов.

ЛАБОРАТОРНАЯ РАБОТА № 4

Разработка функций рекурсивных и без рекурсии, использующих параметр–указатель на функцию

Цель лабораторной работы

Целью лабораторной работы является получение навыков программирования с использованием функций, освоение:

- построения рекурсивных функций;
- правил использования в качестве параметров – указателей на функции;

Теоретические сведения

Рекурсивные функции

Рекурсия – это способ организации обработки данных в функции, когда функция вызывает сама себя прямо или косвенно.

Функция называется косвенно рекурсивной, если содержит обращение к другой функции, которая содержит прямой или косвенный вызов определяемой (первой) функции.

Если в теле функции явно используется вызов этой функции, то имеет место прямая рекурсия.

Рекурсивная форма алгоритма дает более компактный программный текст, но требует дополнительных затрат оперативной памяти - для размещения данных и времени - для рекурсивных вызовов функции.

При выполнении рекурсивной функции происходит ее многократный вызов и при этом:

1) в стеке сохраняются значения всех локальных переменных и параметров функции для всех предыдущих вызовов, выделяется память для локальных переменных очередного вызова;
2) переменным с классом памяти **extern** и **static** память выделяется один раз, которая сохраняется в течение всего времени программы.

Рекурсивный алгоритм позволяет повторять операторы тела функции многократно, каждый раз с новыми параметрами, конкурируя тем самым с итерационными методами. И так же, как и в итерационных методах алгоритм должен включать условие для завершения повторения обработки данных, то есть иметь ветвь решения без рекурсивного вызова функции.

Указатели на функции

Имя функции является указателем-константой, значением которого является адрес размещения операторов функции в оперативной памяти. Это значение адреса можно присвоить указателю-переменной на подобные функции, и затем этот указатель-переменную можно применять для вызова функции.

Указатель на функцию – это некоторая переменная, значениями которой будут являться адреса функций, характеристики которых (тип результата и сигнатура параметров) должны быть указаны в определении указателя.

Определение указателя на функцию:

<тип_функции> (* имя указателя) (спецификация_параметров) =
<имя инициирующей функции>;

Как только некоторому указателю присвоено имя функции, вызов этой функции можно производить как стандартно, так и используя имя указателя на функцию, который хранит адрес этой функции.

Указателю – переменной можно присваивать имена различных функций (указателей – констант), у которых соответствующие указателю тип результата и сигнатура параметров. Присваивая указателю имена (адреса) различных функций, можно организовать вызов той или иной функции в соответствии со значением указателя.

Массивы указателей на функции

Указатели на функции могут быть объединены в массивы.

Ниже дано определение массива указателей на функции, возвращающие значение типа **float** и имеющие два параметра типа **char** и **int**. В массиве с именем **pArray** четыре таких указателя:

float (*pArray) (char, int) [4] ; //массив из 4 –х указателей.

Эквивалентное определение массива **ptrArray**:

float (*ptrArray [4])(char, int);

При объявлении массива можно провести инициализацию элементов массива указателей на функции именами соответствующих функций.

Массивы указателей на функции удобно использовать при разработке программ, управление которыми выполняется с помощью меню, реализующее вызов различных функций обработки данных в интерактивном режиме.

Рассмотрим алгоритм работы простейшего меню:

- все варианты обработки данных определяются в виде функций**;**
- объявляется массив из указателей на функции, инициированный именами функций обработки данных.

Интерактивная часть:

- на экран выводятся строки описания вариантов обработки данных и соответствующие вариантам целочисленные номера;
- пользователю предлагается выбрать из меню нужный ему пункт и ввести значение номера, соответствующее требуемому варианту обработки;
- пользователь вводит значение номера с клавиатуры;
- по номеру пункта, как по индексу, из массива указателей выбирается соответствующий элемент, инициированный адресом нужной функции обработки; производится вызов функции.

Использование массива указателей существенно упрощает программу, так как в данном случае отпадает необходимость использовать оператор **switch** для выбора варианта.

Определение типа указателя на функцию

Для удобства последующих применений целесообразно определить имя типа указателя на функцию с помощью спецификатора **typedef**:

typedef <тип_функции> (*имя типа указателя) (спецификация параметров);

После описания типа можно определять переменные данного типа, например, указатели или массивы указателей.

Указатель на функцию - параметр функции

В С++ все функции внешние, любую определенную функцию можно вызывать в теле любой другой функции непосредственно по имени, не передавая имя функции через механизм параметров.

Указатели на функции как параметры функций целесообразно использовать, когда в создаваемой функции должна быть заложена возможность обработки не конкретной, а произвольной функции. В этом случае адрес обрабатываемой функции целесообразно передавать в функцию посредством параметра.

Указатели на функции удобно использовать в качестве параметров функций, реализующих ту или иную обработку других функций, которые заранее не определенны.

В таких случаях ***формальным*** параметром функции должен быть указатель на передаваемую функцию, а ***фактическим*** параметром должно быть имя передаваемой функции.

Указатели на функции в качестве формальных параметров можно использовать, например, в функциях:

- формирования таблиц результатов, получаемых с помощью различных функций (формул);
- вычисления интегралов с различными подынтегральными функциями;
- нахождения сумм рядов с различными общими членами и т. д.

Задание для выполнение лабораторной работы

Дано:

- аналитическая функция, зависящая от одного вещественного аргумента **x;**
- формула для вычисления значения функции в некоторой точке **x** дана в виде суммы бесконечного степенного ряда, представляющего убывающую последовательность членов, значения которых стремятся к нулю с ростом их номеров;
- точность вычисления значения функции, определяющая пределы суммирования.

Разработать: структуру программы; схемы алгоритмов функций; программу для вычисления значений функции с заданной точностью для заданных значений аргумента и вывода значений аргумента и функции в табличной форме.

Вычисление значений заданной функции, а именно суммы ряда, произвести следующими ***шестью способами***:

1) с помощью встроенной функции (1 способ);

2) для вычисления очередного члена ряда использовать рекуррентную формулу; для суммирования очередных членов ряда использовать:

- оператор **while (**2-й способ);
- оператор **do-while** (3-й способ);

3) для вычисления значения очередного члена ряда определить функцию, возвращающую значение члена ряда, в зависимости от его номера в точке **x**; передать данную функцию посредством параметра в функцию, производящую суммирование членов ряда; для суммирования очередных членов ряда использовать:

- оператор **while** (4-й способ);
- оператор **do-while** (5-й способ);
- рекурсивное суммирование (6-й способ).

Каждый способ вычисления значений исходной функции должен быть оформлен в виде функции, возвращающей результат с помощью оператора **return**.

Порядок выполнения работы

1) вывести рекуррентную формулу для вычисления значения очередного члена ряда через предыдущий:

$$a_{n+1} = f(x,n) \cdot a_n;$$

2) определить функцию **next()**, возвращающую значение очередного члена ряда, в зависимости от его номера **n** в точке **x**;

3) определить тип указателя на функцию, такую как **next().** Тип указателя использовать при определении параметра - указателя на функцию для передачи функции **next()** в функцию, вычисляющую сумму членов ряда;

4) разработать алгоритмы и определить пять функций, вычисляющих различными способами (задание для выполнения лабораторной работы) значение суммы ряда с заданной точностью; суммирование проводить, пока значение очередного члена ряда не станет меньше значения точности;

5) в главной функции задать значение точности, а также диапазон и шаг изменения аргумента функции. Вывести в файл таблицу значений функции, полученных шестью способами.

Пример выполнения лабораторной работы

Задание: Дана функция и ряд для вычисления значения функции:

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots \text{ при } |x| < \infty .$$

Вычислить значения функции с точностью $e=10^{-5}$ в диапазоне значений аргумента x = -2 ÷ 2 с шагом 1.

Рекуррентная формула равна:

$$a_{n+1} = - a_n * x^2 / (2*(n+1)*(2*n+3)).$$

На рис. 2.1 приведена структура программы, на рис. 2.2 – листинг программы, на рис. 2.3 - файл с результирующими данными.

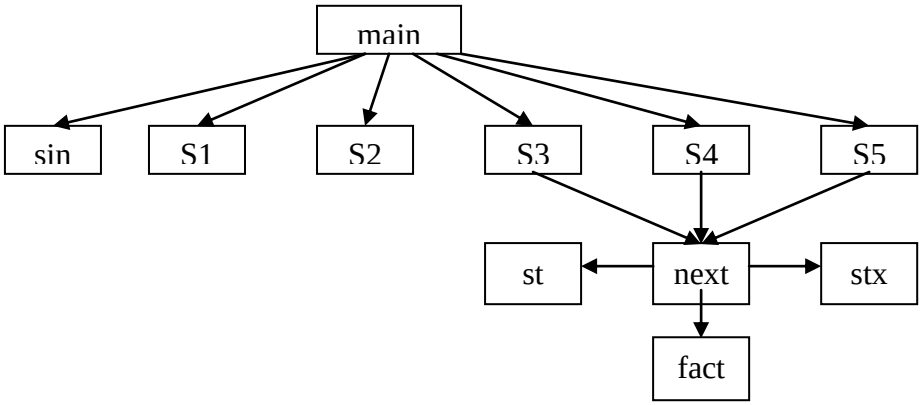

Рис. 2.1. Структура программы

```
//Лабораторная работа №4 студента группы ЭВМ 1-1 Иванова Петра
#include <math.h>
#include <iomanip.h>
#include <fstream.h>
#include <stdlib.h>
ofstream fout; //Объявление выходного файлового потока
//---------------------------------Вычисление факториала-----------------------------
long fact(int n)
{if(n<0) return (0);
 else if(n==0) return (1);
 else return n*fact(n-1);}
//---------------------------------Вычисление x^n ----------------------------------------
double stx ( double x , int n )
{if (n == 0) return (1);              // здесь рекурсия прерывается;
 if (x == 0) return (0);
 if (n==1) return (x);
 if ( n > 0 ) return (x * stx (x, n-1 ));
 if ( n < 0 ) return (stx ( x , n+1 ) / x);
}
//---------------------------------Вычисление (−1)^n ---------------------------------
int st (int n) {
if((n= =0) || (n%2= =0)) return 1;
 else return (-1);
}
//---------------------- Функция вычисления очередного члена ряда---------------
float next ( float x, int n)
{ return st(n)*stx(x,2*n+1)/fact(2*n+1);}
//-------------------------Описание типа – указателя на функции------------------
typedef float (*func)(float, int);
```

Рис. 2.2. Листинг программы

```
//----------Вычисление суммы с помощью while и рекуррентной формулы---
float S1 (int n, float x, float e)
{ float a =x, s=0;
 while (fabs(a)>e)
{s+=a; a*=(-1* x*x/(2*(n+1)*(2*n+3))); n++; }
 return (s+a);
}
//-------Вычисление суммы с помощью do-while и рекуррентной формулы--
float S2 ( int n, float x, float e)
{ float a =x , s=0;
 do
{s+=a; a*=(-1* x*x/(2*(n+1)*(2*n+3))); n++; }
 while (fabs(a)>e);
 return (s+a);
}
//----------Вычисление суммы с помощью while и параметра – функции------
float S3 ( int n, float x, float e, func fn)
{ float f = fn(x,n), s=0;
 while(fabs(f)>e)
 {s+=f; n++; f=fn(x,n);}
 return (s+f);
}
//--------Вычисление суммы с помощью do-while  и параметра – функции----
float S4 ( int n, float x, float e, func fn)
{ float f=fn(x,n) , s=0;
 do
 {s+=f; n++; f=fn(x,n);}
 while(fabs(f)>e);
 return (s+f);
}
//-----С помощью рекурсивного суммирования и параметра – функции-------
float S5 ( int n, float x, float e, func fn)
{ float f = fn(x,n);
 if(fabs(f)<e) return f;
 else return (f+ S5(n+1, x, e, fn));
}
//-----------------------------------Главная функция-----------------------------------
void main()
{float e = 1.e-5;              // точность вычислений;
fout.open("l5.res");          // открытие файла результатов для записи;
if(!fout) { cout<<"Ошибка открытия файла результатов"; exit(0);}
```

Рис. 2.2. (Продолжение)

```
//--------------------------Вывод заголовка и шапки таблицы---------------------
fout<<"\t\t\tРезультаты для e = " <<e<<'\n'
<< "|-----|----------|----------|----------|----------|----------|----------|" <<'\n'
<< "|  x  |    sin   |    S1    |    S2    |    S3    |    S4    |    S5    |" <<'\n'
<< "|-----|----------|----------|----------|----------|----------|----------|";
for(float x=-2;x<=2;x+=1)
{ fout<< endl<<'|'<<setw(4)<<x<<setw(2)<<'|'<<setw(10)<<sin(x)<<'|'
 <<setw(10) << S1(0,x,e)<<'|'<<setw(10) << S2(0,x,e) << '|' << setw(10)
 << S3(0,x,e,next) << '| << setw(10) << S4(0,x,e,next) << '|' < setw(10) <<
S5(0,x,e,next) << '|';
}
fout<<'\n'<<"|-----|----------|----------|----------|----------|----------|----------|";
fout.close();
}
```

Рис. 2.2. (Продолжение)

Результаты для e = 1e-05

x	sin	S1	S2	S3	S4	S5
-2	-0.909297	-0.909297	-0.909297	-0.9093	-0.909305	-0.9093
-1	-0.841471	-0.841471	-0.841471	-0.841471	-0.841474	-0.841471
0	0	0	0	0	0	0
1	0.841471	0.841471	0.841471	0.841471	0.841474	0.841471
2	0.909297	0.909297	0.909297	0.9093	0.909305	0.9093

Рис. 2.3. Содержимое файла результатов l5.res

Контрольные вопросы

1. Что может возвращать функция с помощью указателя в **return?**
2. Как можно возвратить из функции с помощью оператора **return** одномерный или двумерный динамический массив?
3. Что такое рекурсивная функция, какова ее структура?
4. Указатели на функции (определение указателя, массива указателей, определение типа указателя на функцию).
5. Как указатели на функции используются как параметры функций?
6. Что такое рекуррентная формула?

Варианты заданий лабораторной работы

1. $arctg(X) = \sum_{n=0}^{\infty}(-1)^n \cdot \frac{X^{2n+1}}{2n+1} = X - \frac{X^3}{3} + \frac{X^5}{5} - \frac{X^7}{7} + \ldots$, при $|X| < 1$;

2. $arctg(X) = \pm\frac{\pi}{2} + \sum_{n=0}^{\infty}(-1)^{n+1} \cdot \frac{1}{(2n+1)\cdot X^{2n+1}} =$

$= \pm\frac{\pi}{2} + (-\frac{1}{X} + \frac{1}{3X^3} - \frac{1}{5X^5} + \ldots)$, при $|X| > 1$;

$\pi/2$ берется со знаком «+» при $X>1$ и со знаком «-» при $X<1$;

3. $arcctg(X)=\frac{\pi}{2}-arctg(X)=\frac{\pi}{2}-\sum_{n=0}^{\infty}(-1)^n\cdot\frac{X^{2n+1}}{2n+1}=\frac{\pi}{2}-(X-\frac{X^3}{3}+...)$, при $|X|<1$

4. $\arcsin(X)=arctg\left(\frac{X}{\sqrt{1-X^2}}\right)=X+\sum_{n=1}^{\infty}\frac{1\cdot3\cdot5\cdot...\cdot(2n-1)\cdot X^{2n+1}}{2\cdot4\cdot6\cdot...\cdot(2n)\cdot(2n+1)}=$

$=X+\frac{X^3}{2\cdot3}+\frac{1\cdot3\cdot X^5}{2\cdot4\cdot5}+\frac{1\cdot3\cdot5\cdot X^7}{2\cdot4\cdot6\cdot7}+...,$ при $|X|<1;$

5. $\arccos(X)=\frac{\pi}{2}-arctg\left(\frac{X}{\sqrt{1-X^2}}\right)=\frac{\pi}{2}-X-\sum_{n=1}^{\infty}\frac{1\cdot3\cdot5\cdot...\cdot(2n-1)\cdot X^{2n+1}}{2\cdot4\cdot6\cdot...\cdot(2n)\cdot(2n+1)}=$

$=\frac{\pi}{2}-X-\left(\frac{X^3}{2\cdot3}+\frac{1\cdot3\cdot X^5}{2\cdot4\cdot5}+\frac{1\cdot3\cdot5\cdot X^7}{2\cdot4\cdot6\cdot7}+...\right),$ при $|X|<1;$

6. $\cos(X)=\sum_{n=0}^{\infty}(-1)^n\cdot\frac{X^{2n}}{(2n)!}=1-\frac{X^2}{2!}+\frac{X^4}{4!}-\frac{X^6}{6!}+...,$ при $|X|<\infty;$

7. $\sin^2(X)=\sum_{n=1}^{\infty}(-1)^{n+1}\cdot\frac{2^{2n-1}X^{2n}}{(2n)!}=\frac{2^1\cdot X^2}{2!}-\frac{2^3\cdot X^4}{4!}+\frac{2^5\cdot X^6}{6!}-...,$ при $X<1;$

8. $\cos^2(X)=1-\sum_{n=1}^{\infty}(-1)^{n+1}\cdot\frac{2^{2n-1}X^{2n}}{(2n)!}=1-(\frac{2^1\cdot X^2}{2!}-\frac{2^3\cdot X^4}{4!}+\frac{2^5\cdot X^6}{6!}-...)$, при $X<1;$

9. $\sin^3(X)=\frac{1}{4}\cdot\sum_{n=1}^{\infty}(-1)^{n+1}\cdot\frac{3^{2n+1}-3}{(2n+1)!}\cdot X^{2n+1}=\frac{1}{4}\cdot(\frac{3^3-3}{3!}\cdot X^3-\frac{3^5-3}{5!}\cdot X^5+..)$, при $X<1;$

10. $\cos^3(X)=\frac{1}{4}\cdot\sum_{n=0}^{\infty}(-1)^n\cdot\frac{3^{2n}+3}{(2n)!}\cdot X^{2n}=\frac{1}{4}\cdot(\frac{3^0+3}{0!}\cdot X^0-\frac{3^2+3}{2!}\cdot X^2+...)$, при $X<1;$

11. $e^X=\sum_{n=0}^{\infty}\frac{X^n}{n!}=1+\frac{X^1}{1!}+\frac{X^2}{2!}+\frac{X^3}{3!}+...,$ при $|X|<\infty;$

12. $e^X\cdot(1+X)=\sum_{n=0}^{\infty}\frac{(n+1)\cdot X^n}{n!}=1+\frac{2\cdot X^1}{1!}+\frac{3\cdot X^2}{2!}+\frac{4\cdot X^3}{3!}+...$, при $|X|<=2{,}4;$

13. $e^{-X^2}=\sum_{n=0}^{\infty}(-1)^n\frac{X^{2n}}{n!}=1-\frac{X^2}{1!}+\frac{X^4}{2!}-\frac{X^6}{3!}+...$, при $X<1;$

14. $\ln X=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{(X-1)^n}{n}=\frac{(X-1)^1}{1}-\frac{(X-1)^2}{2}+\frac{(X-1)^3}{3}-...$, при $0<X<=2;$

15. $\ln(1+X)=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{X^n}{n}=X-\frac{X^2}{2}+\frac{X^3}{3}-\frac{X^4}{4}+...,$ при $-1<X<=1;$

16. $\ln(1-X)=-\sum_{n=1}^{\infty}\frac{X^n}{n}=-(X+\frac{X^2}{2}+\frac{X^3}{3}+\frac{X^4}{4}+...),$ при $X<1;$

17. $\ln\left(\frac{1+X}{1-X}\right)=2\cdot\sum_{n=0}^{\infty}\frac{X^{2n+1}}{(2n+1)}=2\cdot(X+\frac{X^3}{3}+\frac{X^5}{5}+\frac{X^7}{7}+...)$, при $|X|<1;$

18. $\ln\left(\frac{X+1}{X-1}\right)=2\cdot\sum_{n=0}^{\infty}\frac{1}{(2n+1)\cdot X^{2n+1}}=2\cdot(\frac{1}{X}+\frac{1}{3X^3}+\frac{1}{5X^5}+\frac{1}{7X^7}+...)$, при $|X|>1;$

19. $\ln X = 2 \cdot \sum_{n=0}^{\infty} \frac{(X-1)^{2n+1}}{(2n+1)\cdot(X+1)^{2n+1}} = 2 \cdot (\frac{X-1}{X+1} + \frac{(X-1)^3}{3(X+1)^3} + ...),$ при $X > 0;$

20. $\ln X = \sum_{n=1}^{\infty} \frac{(X-1)^n}{n \cdot X^n} = \frac{X-1}{X} + \frac{(X-1)^2}{2X^2} + \frac{(X-1)^3}{3X^3} + ...,$ при $X > 0{,}5;$

21. $sh(x) = \frac{e^X - e^{-X}}{2} = \sum_{n=1}^{\infty} \frac{X^{2n-1}}{(2n-1)!} = X + \frac{X^3}{3!} + \frac{X^5}{5!} + \frac{X^7}{7!} + ...,$ при $|X| < \infty;$

22. $ch(x) = \frac{e^X + e^{-X}}{2} = \sum_{n=0}^{\infty} \frac{X^{2n}}{(2n)!} = 1 + \frac{X^2}{2!} + \frac{X^4}{4!} + \frac{X^6}{6!} + ...,$ при $|X| < \infty;$

23. $arsh(X) = \ln(X + \sqrt{X^2+1}) = X + \sum_{n=1}^{\infty} (-1)^n \frac{1 \cdot 3 \cdot 5 \cdot ... \cdot (2n-1) X^{2n+1}}{2 \cdot 4 \cdot 6 \cdot ... \cdot (2n) \cdot (2n+1)} =$

$= X - \frac{1 \cdot X^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot X^5}{2 \cdot 4 \cdot 5} - \frac{1 \cdot 3 \cdot 5 \cdot X^7}{2 \cdot 4 \cdot 6 \cdot 7} + ...,$ при $|X| < 1;$

24. $arch(X) = \ln(X + \sqrt{X^2-1}) = \ln(2X) - \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot 5 \cdot ... \cdot (2n-1)}{2 \cdot 4 \cdot 6 \cdot ... \cdot (2n) \cdot (2n) X^{2n}} =$

$= \ln(2X) - \frac{1}{2 \cdot 2 \cdot X^2} - \frac{1 \cdot 3}{2 \cdot 4 \cdot 4 \cdot X^4} - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 6 \cdot X^6} - ...,$ при $X > 1;$

25. $arcth(X) = \frac{1}{2} \cdot \ln\left(\frac{1+X}{1-X}\right) = \sum_{n=0}^{\infty} \frac{X^{2n+1}}{2n+1} = X + \frac{X^3}{3} + \frac{X^5}{5} + \frac{X^7}{7} + ...,$ при $|X| < 1;$

26. $arcth(X) = \frac{1}{2} \ln\left(\frac{X+1}{X-1}\right) = \sum_{n=0}^{\infty} \frac{1}{(2n+1) \cdot X^{2n+1}} = \frac{1}{X} + \frac{1}{3X^3} + \frac{1}{5X^5} + ...,$ при $|X| > 1;$

ЛАБОРАТОРНАЯ РАБОТА № 5

Разработка программ с использованием структурированных данных

Цель лабораторной работы

Целью лабораторной работы является освоение:

- правил объявления и работы с переменными структурных типов: структур, массивов структур, указателей на структуры;
- правил обращения к элементам структур и элементам массива структур;
- ввода и вывода значений элементов массива структур;
- методов форматирования вывода данных в текстовой файл;
- алгоритма поиска данных в массиве структурированных данных.

Теоретические сведения.

Структурные типы и структуры

Структуры – это данные, которые объединяют в единое целое множество возможно разнотипных именованных элементов**.**

Элементы структуры могут быть одного или разных типов, но все они должны иметь различные имена.

Компонентами (элементами, членами, полями) структуры могут быть переменные как простых базовых, так и производных типов. Элементами структур могут быть, например, массивы данных, а также другие структуры, которые в этом случае называются вложенными.

Перед определением структуры должен быть описан ее структурный тип, который и задает внутреннее строение структуры:

struct <имя типа> {список описаний членов структуры через ‘;’ };

Определение структуры:

<имя типа> <имя структуры>;

Для обращения к элементам, входящим в состав структуры, используются уточненные имена**:**

<имя структуры>.<имя элемента>

Можно объявлять, указатели на структуры:

<имя структурного типа> * <указатель на структуру>;

Значением указателя на структуру может быть адрес структуры того же типа. После объявления указателя, который владеет адресом некоторой структуры, к элементам структуры можно обращаться двумя способами:

1) используя операцию **‘->’** доступа к элементам структуры, с которой в этот момент связан указатель:

имя указателя -> имя элемента структуры

2) используя операцию разыменования указателя и формирования уточненного имени:

(*имя указателя). имя элемента структуры

Задание на выполнение лабораторной работы

1. Даны сведения о данных, обладающих несколькими разнотипными характеристиками;
2. Создать текстовой файл данных, в который поместить данные;
3. Определить структурный тип для представления данных;
4. Разработать структуру программы, схемы алгоритмов и программу обработки структурированных данных конкретного варианта.

Порядок выполнения работы

1. Сформировать файл с исходными данными в соответствии с вариантом задания. В каждой строке файла должны располагаться записи характеристик одного из объектов данных.
2. Определить в соответствии с данными структурный тип.
3. Определить внешний массив структур данного типа.
4. Разработать алгоритмы и определить функции:

- подсчета количества записей (строк) в файле данных;
- удаления пробелов в начале и в конце строки;
- ввода данных файла в массив структур;
- вывода шапки таблицы;
- вывода данных массива структур в файл результатов в виде таблицы;
- поиска и вывода элементов массива структур, поиск производить по одному признаку.

5. В главной функции произвести вызов функций.
6. Файл результатов должен содержать таблицу структурных данных, а также результаты поиска структур.

Пример выполнения лабораторной работы

Даны сведения об автомобилях: марка, цвет, цена, страна изготовитель, максимальная скорость. Исходные данные для создания массива структур поместить в файл данных в виде, представленном на рис. 3.1. Исходные данные для поиска по марке автомобиля показаны на рис. 3.2 (файл poisk.dat). Листинг программы и файл результатов представлены на рис. 3.3. и 3.4 соответственно.

```
Reno        white   15000     France     250
Fiat        green    5000     Italy      200
Ford SCP    braun   10000     USA        230
Mercedec    blue    30000     German     300
Volga       black    9000     Rusia      180
```

Рис. 3.1. Файл данных

```
Mercedec
Ford SCP
Fiat

Volga
Oka
```

Рис. 3.2. Файл данных для поиска poisk.dat

```
//Лабораторная работа №5 студента группы ЭВМ 1-1 Иванова Петра
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <iomanip.h>
#include <fstream.h>
```

Рис. 3.3. Листинг программы

```
struct avto {char mark [20];          //марка автомобиля;
             char color [15];         //цвет;
             double price;            //цена;
             char place [15];         //страна изготовитель;
             unsigned int spid;       //максимальная скорость;
             };
avto B[10];        //объявляется массив из 10 структур типа avto
char*sh[6]={
"                  Легковые автомобили                                  ",
"╔═══════════════╤═══════╤═══════╤═══════════════╤═══════════════╗",
"║Марка          │Цвет   │Стоимость│Изготовитель  │Скорость       ║",
"║               │       │         │               │максимальная   ║",
"╟═══════════════┼═══════┼═══════┼═══════════════┼═══════════════╢",
 };
ifstream fin ;              // определяется входной файловый поток;
ofstream fout;              //определяется выходной файловый поток;
double s;
//------------удаление пробелов вначале и в конце строки - параметра--------
void filtr(char *Stroka)
{ char Source[255]="";          // вспомогательная строка;
 int Xl=0, X2=strlen(Stroka)-1;       //установка на начало и конец строки;
 while (Stroka[Xl] ==' ') X1++;
 while (Stroka[X2] ==' ' || Stroka[X2]= ='\n') X2--;
 for(int x =Xl; x<=X2 ; x++)
 Source[x-Xl]=Stroka[x];
 Source[x-X1]='\0';       // в конце строки устанавливаем байтовский ноль;
strcpy(Stroka,Source);    //строку Source копируем в массив Stroka;
}
//---------------------Вывод строк шапки таблицы-------------------------
void psh() {
 for(int i=0;i<5; i++)
 fout<<sh[i]<<endl;
}
//------------Определение количества записей в файле данных--------------
int number (char* filename)     //параметр – строка с именем файла данных
{ int n=0;                      // счетчик записей;
 char T[255];                   // массив буфер;
 fin.open(filename );           //открытие файла данных;
 if (!fin) {cout<<"Ошибка при открытии файла данных"; exit(0);}
 while(!fin.eof())              //пока не конец файла данных
{ fin.getline(T,255); n++;}
```

Рис. 3.3. (Продолжение)

```
 fin.close();                    // закрываем файл данных;
 return n;               //возвращаем количество строк;
}
//------------------Ввод данных из файла и заполнение массива структур-------
void vvod (avto B [10], int n, double & s, char*filename)
{int i; s=0;
 char T[80];                                //массив буфер;
 fin.open(filename );                       //открытие файла данных;
 if (!fin) {cout<<"Ошибка при открытии файла данных"; exit(0);}
 for(i =0; i< n; i++ )
{ fin.getline(B[i].mark,14) ;filtr(B[i].mark);//считывается марка автомобиля;
 fin>>B[i].color >> B[i].price;                    //считывается цвет и цена;
 fin.getline (B[i].place,15); filtr (B[i].place);          // страна изготовитель;
 fin>>B[i].spid;                            //максимальная скорость;
 s+=B[i].price;       // подсчитывается суммарная цена всех автомобилей;
 fin.getline(T,80);}//из потока извлекается символ перехода на новую строку
fin.close();}
//------------------------------Вывод  массива  структур-------------------------------
void vivod (avto B[10], int n)
 { int i;
 for( i=0; i<n; i++)
 fout<<'\272'<<setw(14)<<B[i].mark<<'\272'<<setw(8)<<B[i].color<<'\272'
 <<setw(9)<<B[i].price<< '\272'<<setw(14)<<B[i].place<< '\272'<<setw(13)
 <<B[i].spid<<'\272'<<endl;
 fout<<sh[5]<<endl<<"Суммарная стоимость автомобилей: "<<s; }
//----------------------Поиск по марке автомобиля--------------------------------
void poisk (avto *p, int n)
{char name[20];                    // строка для поискового признака;
 fin.open("poisk.dat" );            //открытие файла данных для поиска;
 if (!fin) {cout<<"Ошибка открытия файла данных для поиска "; exit(0);}
fout<<"\n\n Поиск по марке автомобиля \n";
while(!fin.eof())                  // пока не конец файла;
 {fin.getline(name,20);            //считываем из файла поисковый признак;
 if(!strcmp(name,""))              //если это пустая строка
 {fout<<"\nНет данных"; continue;} //к следующей итерации цикла while;
//в противном случае
 for(int i=0;i<n;i++)              //перебираем структуры из массива структур
if(strcmp(p[i].mark,name)==0) //если совпали
  {fout<<"\n\272"<<setw(10)<<p[i].mark<<'\272'<<setw(8)<<p[i].color<<'\272'
 <<setw(9)<<p[i].price<< '\272'<<setw(14)<<p[i].place<< '\272'<<setw(13)
 <<p[i].spid<<'\272';goto m;}    //тело цикла for
//если перебрали все структуры массива и совпадения не было
 fout<<"\n Марка "<< name<<" не найдена";
```

```
 m:; } //тело цикла while
fin.close(); }
//-----------------------Главная функция---------------------------------------
 void main() {char filen [15];
 cout<<"Введите имя файла данных - ";
 cin.getline(filen,80);              // ввод имени файла;
 fout.open("res.cpp" );              // открытие файла результатов;
 if (!fout) {cout<<"Ошибка открытия файла результатов"; exit(0);}
 //вызовы функций:
 int n=number(filen);                //количество записей в файле;
 vvod(B,n,s,filen); psh(); vivod(B,n); poisk(B,n);}
```

Рис. 3.3. (Продолжение)

```
                    Легковые автомобили
```

Марка	Цвет	Стоимость	Изготовитель	Скорость максимальная
Reno	white	15000	France	250
Fiat	green	5000	Italy	200
Ford SCP	braun	10000	USA	230
Mercedec	blue	30000	German	300
Volga	black	9000	Rusia	180

```
Суммарная стоимость автомобилей: 69000

Поиск по марке автомобиля

|| Mercedec||   blue||  30000||       German||        300||
|| Ford SCP||  braun||  10000||          USA||        230||
||     Fiat||  green||   5000||        Italy||        200||
 Нет данных для поиска
||    Volga||  black||   9000||        Rusia||        180||
 Марка Oka не найдена
```

Рис. 3.4. Файл результатов res.cpp

Контрольные вопросы

1. Как определять структуру, массив структур, указатель на структуру?
2. Какой объем оперативной памяти получает структура, массив структур?
3. Динамическое выделение памяти на структуру, массив структур.
4. Как обращаться к полям структуры или элемента массива структур?
5. Как обращаться к полям структуры или элемента массива структур, если объявлен указатель на структуру или указатель на элемент массива структур?
6. Структура – параметр функции (передача параметра по значению, по адресу, по ссылке).
7. Данные структурного типа – результат выполнения функций с помощью оператора **return** и с помощью формального параметра.

8. Что может быть фактическим параметром функции, если формальный параметр задан в виде указателя на структурный тип?
9. Поясните назначение определения char*sh[6]. Какой объем памяти и значение получает переменная sh?

Варианты заданий лабораторной работы

1. Студенты факультета

Номер зачетной книжки	Название группы	Фамилия, инициалы	Размер стипендии	Средний вступительный бал

Определить средний балл студентов факультета.

2. Сотрудники отдела

Табельный номер	Фамилия, инициалы	Дата рождения	Оклад в тыс. руб.	Стаж работы

Определить фонд заработной платы отдела.

3. Города

Название	Численность населения	Страна	Год создания	Площадь в кв. км

Определить город с максимальной численностью.

4. Туристические маршруты агентства

Наименование маршрута	Страна	Длительность тура	Стоимость	Срок (число,месяц,год)

Определить суммарную стоимость туров.

5. Самолеты

Название типа	Фамилия конструктора	Год создания	Количество мест	Грузоподъемность в тоннах

Определить суммарную грузоподъемность самолетов.

6. Перевозки

Тип самолета	Номер борта	Количество рейсов	Налет в часах	Налет в тыс. км

Определить суммарный налет в часах.

7. Расписание

Номер рейса	Наименование рейса	Тип самолета	Стоимость билета	Время полета в часах

Определить минимальную и максимальную стоимость билетов.

8. Экскурсии

Наименование	Страна	Продолжительность	Стоимость	Транспорт

Определить самую продолжительную экскурсию.

9. Спортивные секции

Название секции	ФИО тренера	Длительность тренировки	Стоимость одного занятия	Количество спортсменов

Определить общее количество спортсменов по всем секциям.

10. Музеи

Название музея	Характер экспозиций	Адрес	Стоимость билета	Время работы

Определить минимальную и максимальную стоимость билетов.

11. Кинотеатры

Наименование кинотеатра	Адрес	Время сеансов	Стоимость билета	Количество мест

Определить минимальную стоимость билетов.

12. Линии метро

Наименование	Район линии	Год пуска	Протяженность в км	Количество поездов

Определить общую протяженность линий метро.

13. Книги

Название книги	ФИО автора	Шифр	Издательство	Стоимость

Определить общую стоимость книг.

14. Кассы аэропорта

Номер кассы	ФИО кассира	Дата продаж	Выручка	Количество проданных билетов

Определить общую выручку.

15. Телевизоры на складе

Наименование	Фирма изготовитель	Стоимость	Размер экрана	Количество на складе

Определить общее количество телевизоров.

16. Ведомость зарплаты предприятия

ФИО	Название отдела	Табельный номер	Количество раб. часов	Размер зарплаты

Определить среднюю зарплату по предприятию.

17. Мосты города

Наименование	Страна	Высота	Протяженность	Количество опор

Определить максимальную протяженность мостов.

18. Квартиры

Адрес квартиры	Площадь в кв. м	Сторона света	Этаж	Количество комнат	Стоимость 1 кв. м

Определить среднюю стоимость 1 кв. м

19. Склад товаров

Наименование товара	Артикул товара	Фирма изготовитель	Стоимость одной единицы	Количество на складе

Определить самый дорогой товар склада.

20. Холодильники на складе

Наименование	Фирма изготовитель	Стоимость	Емкость камеры	Количество на складе

Определить общее количество холодильников на складе.

21. Абонентская плата за телефон

ФИО абонента	Номер телефона	Год установки	Количество абонентов	Плата за телефон

Определить среднюю плату за телефон.

22. Продажа программных продуктов

Наименование продукта	Фирма разработчик	Стоимость в тыс. руб.	Объем в Мбайтах	Количество на складе

Определить суммарную стоимость программных продуктов.

23. Супермаркеты

Наименование	Адрес	Этажность	Характер продаж	Средняя дневн. выручка

Определить самый доходный супермаркет.

24. Характеристики персональных компьютеров

Тип процессора	Тактовая частота	Емкость ОП в Мб	Емкость ЖД в Гб	Тип монитора

Определить максимальную емкость ОП.

25. Детские сады

Наименование детского сада	Район города	Категория	Оплата за месяц	Количество детей

Определить среднюю по всем детским садам оплату за месяц.

ЛАБОРАТОРНАЯ РАБОТА № 6
Обработка данных текстовых и бинарных файлов

Цель лабораторной работы

Целью лабораторной работы является получение навыков программирования с использованием текстовых и бинарных файлов, содержащих структурированные данные.

Теоретические сведения.

Работа с файлами

Основное отличие внешней памяти компьютера от основной (иначе оперативной) памяти состоит в возможности сохранения информации при

отключении компьютера. Информация во внешней памяти (на жестком диске, на магнитных лентах, на оптическом диске на дискетах и т.д.) сохраняется в виде файлов, именованных объектов внешней памяти, доступ к которым поддерживается операционной системой.

Все что надо – задать способ связи программы с файлом, а также иметь функции, используемые программой при чтении содержимого файла, записи в файл, создания нового файла, позиционирования записи и чтения данных в файл и из файла.

Такие действия являются частью аспекта **ввода/вывода** данных и для их реализации в С++ имеются различные средства**.**

Все средства имеют альтернативные варианты методов выполнения основных работ с файлами, здесь будут рассмотрены некоторые методы библиотеки потоковых классов.

Текстовые и бинарные (двоичные) файлы

Когда данные сохраняются в файле, их можно сохранять как в текстовом, так и в двоичном формате.

Текстовая форма означает, что все данные (даже числа) сохраняются как текст – последовательность символов.

Двоичный формат означает, что данные в файле сохраняются во внутреннем представлении этих данных компьютером.

Для символа двоичное представление совпадает с его текстовым – двоичным представлением ASCII-кода символа.

Однако для чисел двоичное представление очень сильно отличается от их текстового представления.

Рассмотрим пример - форму сохранения значения переменной **float a=0.375;** в текстовом и бинарном файле.

При записи в текстовой файл операция вставки << производит преобразование внутренних кодов переменной в коды символов числа, то есть в файле сохраняются коды следующих символов:

‘0’	‘.’	‘3’	‘7’	‘5’
00110000	00101110	00110011	00110111	00110101

---------- всего 40 бит ----------

Двоичное представление значения переменной в бинарном файле идентичное внутреннему представлению компьютером данной переменной в формате с нормализованной мантиссой и порядком:

Биты двоичной нормализованной мантиссы (в данном случае - правильной дроби)			Биты показателя экспоненты	Бит знака числа
00000000	00000000	00000011	0111110	0

----------всего 32 бита для типа **float**----------

Каждый формат имеет свои достоинства.

Текстовой формат прост для чтения и редактирования файла. Текстовой файл легко переносится с одной компьютерной системы на другую.

В двоичном файле числа сохраняются более точно, поскольку он позволяет сохранить внутреннее представление числа, не происходит ошибок преобразования или округления чисел. Сохранение данных в бинарном файле происходит быстрее, т.к. при этом не происходит преобразования и данные можно сохранять крупными блоками. Однако при переносе данных в другую систему возможны проблемы, если в новой системе применяется иное внутреннее представление данных.

Текстовой файл - это последовательность символьных строк переменной длины, разделенных комбинацией символов CR - “перевод каретки в начало” (символ с кодом 13) и символ LF - “перевод строки” (символ с кодом 10). Как правило, работа с **текстовым файлом** организуется построчно. При записи данных из оперативной памяти в файл значения числовых типов будут преобразовываться из внутренних кодов хранения данных в символьное представление и в таком виде записываться в строку файла. При чтении данных из файла в оперативную память часть строки будет пониматься как символьное представление числовой переменной и при вводе данных в оперативную память выполняется преобразование символов в двоичные коды внутреннего представления данных. Кроме того, в **текстовом режиме** при чтении из файла два символа CR и LF преобразуются в один символ новой строки ‘\n’. При записи в текстовой файл один символ – символ новой строки ‘\n’ преобразуется в два символа CR и LF.

Бинарный файл предназначен для двоичного режима обмена данными, когда преобразование символов не происходит, и их значения не анализируются. Бинарный файл – это линейная последовательность байтов, соответствующая внутреннему представлению данных.

Потоковый ввод/вывод на базе библиотеки классов

Чтобы использовать классы входных/выходных файловых потоков, в программу необходимо включить заголовочный файл **fstream.h**. После этого для работы с файлами требуется выполнить описанные ниже действия:

Создание потоков и открытие файлов

Потоки для работы с файлами являются объектами следующих классов:

ofstream – класс выходных файловых потоков, для организации записи данных в файл;

ifstream – класс входных файловых потоков**,** для чтения данных из файла;
fstream – класс двунаправленных файловых потоков, для организации чтения и записи данных.

Определять конкретные файловые потоки можно, например, так:
ofstream fout; //выходной файловый поток;
ifstream fin; //входной файловый поток;
fstream fio;//входной/выходной (двунаправленный) файловый поток.

Создав файловый поток, нужно присоединить его к конкретному физическому файлу с помощью *компонентной функции* файловых потоков **open().**

Эта функция открывает файл (если он существует) или создает новый файл и связывает его с потоком.

Прототип функции:

void open (const char* filename, int mode= умалчиваемые значения, int protection= умалчиваемые значения);

Первый параметр **filename** – имя уже существующего или вновь создаваемого файла.

Второй параметр **mode** - дизъюнкция флагов, определяющих режим работы с файлом:

ios :: in = 0x01 // открыть только для чтения;
ios :: out = 0x02 // открыть только для записи;
ios :: ate = 0x04 **//** при открытии искать конец файла;
ios :: app = 0x08 **//** дописывать данные в конец файла;
os :: trunc = 0x10 //вместо существующего, создать новый файл;
ios :: nocreate = 0x20 // не открывать новый файл;
ios :: noreplace = 0x40 // не открывать существующий файл;
ios :: binary = 0x80 // открыть файл для двоичного обмена.

Умалчиваемое значение параметра **mode** для потока класса **ifstream** равно **ios::in,** а для потока класса **ofstream** равно **ios::out.**

Третий параметр – **protection** - определяет защиту и достаточно редко используется и, как правило, умалчиваемое значение устраивает пользователя.

Вызов компонентной функции осуществляется с помощью уточненного имени:

имя потока . open(имя файла, режим, защита);

Для проверки удачности завершения функции **open()** используется операция логического отрицания (**!)**. Если ошибок не было, то выражение **!имя потока** имеет нулевое значение.

Закрытие файла

Компонентная функция **close()**, которую наследуют все три файловых потоковых класса, позволяет отсоединить файловый поток от физического файла:

имя файлового потока (присоединенного к файлу).close();

Средства обмена данными с потоком

1. Операции ввода (>>) и вывода (<<) данных.

Операции ввода и вывода определены так, что при той же форме они выполняются по-разному в зависимости от типа правого операнда.

Форма операции вывода << :**поток << выражение**

Форма операции ввода >> :**поток >> l-value.**

При применении операций ввода и вывода к стандартным или файловым потокам по умолчанию устанавливаются стандартные форматы внешнего представления пересылаемых данных.

Например, при выводе данные занимают ровно столько позиций, сколько надо для их представления.

Форматы представления могут быть изменены программистом с помощью:

- флагов форматирования класса ios;
- компонентных функций для управления форматами класса ios;
- с помощью манипуляторов потоков без параметров и с параметрами.

Подробнее операции ввода и вывода данных описаны в лекциях (Ввод/вывод данных – часть 2).

2. Функции двоичного ввода/вывода данных.

Функции вывода

В классе выходного потока определены две функции для двоичного вывода данных в стандартный выходной поток **put()** и **write(),** причем вторая функция перегружена; может иметь два варианта параметров.

Прототипы функций:

ostream & ostream :: put(char c);
ostream & ostream :: write (const signed char * array, int n);
ostream & ostream :: write (const unsigned char * array, int n);

Функция **put()** помещает в выходной поток символ – параметр, при этом следует помнить, что это компонентная функция потока, поэтому вызов ее требует уточненного имени функции:

имя потока.вызов функции, например, **cout.put('X').**

Первый параметр функции **write() -** указатель **array** на участок оперативной памяти, из которого извлекаются побайтно данные для вывода в поток, этот участок трактуется как символьный массив, второй

параметр - **n** определяет количество байт, которые будут выведены в поток.

Функции чтения

Рассмотрим функции бесформатного двоичного чтения класса istream. Вызов такой функции требует уточненного имени:

имя потока . имя функции (список параметров);

Данные считываются в оперативную память без преобразования. Любая информация во входном потоке воспринимается как последовательность байт, читаемая только в символьный массив.

Рассмотрим функции двоичного чтения.

1) **istream & get(signed char * array, int n, char = '\n');**
2**) istream & get (unsigned char * array, int n , char = '\n');**

Функции извлекают последовательность **n-1** байтов из потока и переносят их в символьный массив (буфер), задаваемый первым параметром. Затем в массив помещается концевой символ **'\0'**.

Если раньше в потоке встретился ограничительный символ – третий параметр, то чтение прекращается, сам ограничитель не извлекается из потока, в формируемую строку не переносится, а помещается концевой символ **'\0'**. По умолчанию третий параметр имеет значение **'\n'** – переход на следующую строку, однако при вызове функции ему можно задать другое значение.

Таким образом, формируемый массив должен иметь длину не менее **n** символов. Если встретился символ конца файла (**EOF)**, то он воспринимается как ограничительный символ.

3) **istream & get (signed char & c);**
4) **istream & get (unsigned char & c);**

Функции извлекают один символ из потока и присваивают его параметру.

5) **int get( );**

Функция извлекает символ из потока и возвращает код символа. Если поток пуст, возвращает код конца файла (символ **EOF**, код **–1**).

6) **istream & getline (signed char * array, int n , char= '\n');**
7) **istream & getline (unsigned char * array, int n , char = '\n');**

Функции аналогичны первым двум функциям, но из входного потока читается и символ-ограничитель, однако в массив **array** он также не помещается, а добавляется "концевой" символ строки **'\0'**.

8) **istream & read (signed char * array, int n);**
9) **istream & read (unsigned char * array, int n);**

Функции выполняют чтение из потока заданного количества **n** символов в массив **array.**

10) **int peek( );**

Функция возвращает код очередного символа входного потока (или **EOF**, если поток пуст).

11) **istream & putback(char) ;**

Функция помещает символ во входной поток, этот символ и будет следующим читаемым символом потока.

12) **istream & ignore (int n = 1, int m = EOF);**

Функция позволяет проигнорировать **n** символов входного потока (по умолчанию один символ)**, m -** символ ограничитель, при его появлении выполнение функции прекращается, даже если не все **n** символов извлечены из потока. По умолчанию этот символ равен символу конца файла.

13) **istream & seekg (long pos);**

Функция устанавливает позицию чтения в положение, определяемое значением параметра.

14) **istream & seekg (long pos, seek_dir dir);**

Функция выполняет перемещение позиции чтения от позиции, определяемой вторым параметром **dir**, причем величину и направление перемещения определяет первый параметр **pos.** Параметр **dir** принимает значение из перечисления: **enum seek_dir {ios :: beg, ios :: cur, ios :: end},**
где **beg** – начало потока, **cur** – текущая позиция **, end –** конец потока. Параметр **pos -** величина смещения (в байтах) указателя чтения во входном потоке относительно этих трех ориентиров.

15) **long tellg( );**

Функция возвращает текущую позицию чтения (номер байта в файле).

16) **ostream & seekp(long pos);**

17) **ostream & seekp(long pos, ios :: beg);**

Функции аналогичные функциям 13 и 14, но устанавливают позицию записи в выходном потоке.

18) **long tellp( );**

Функция возвращает текущую позицию записи (номер байта в файле).

Полезные функции:

Функции описаны в файлах **io.h** и **stdio.h:**

int remove (const char*filename);

Функция удаляет существующий на диске файл с именем **filename,** который перед удалением должен быть закрыт, то есть не связан в программе ни с каким файловым потоком. Функция возвращает **0** , если удаление прошло успешно и **–1** , если нет.

int rename (const char * oldname , const char * newname)

Функция переименовывает существующий файл с именем **oldname,** который перед переименованием должен быть закрыт (не связан ни с каким файловым потоком). Новое имя **newname** должно быть оригинально на диске. Функция возвращает **0** , если переименование прошло успешно и **–1** , если нет.

Задание на выполнение лабораторной работы

1. Разработать структуру программы, схемы алгоритмов и программу обработки данных бинарного файла. Файл должен содержать структурированные данные конкретного варианта лабораторной работы. Использовать данные варианта лабораторной работы № 5.
2. Программа должна включать следующие функции обработки данных:
   - создание бинарного файла из текстового файла с данными;
   - дополнение файла новыми записями;
   - чтение данных бинарного файла;
   - поиск структур бинарного файла, поиск производить
     а)по одному поисковому признаку;
     б)по любому сочетанию двух заданных поисковых признаков;
   - модификация ряда структур бинарного файла.
3. Главная функция должна производить вызов разработанных функций.

Порядок выполнения работы

1. Сформировать файлы с данными для тестирования программы:
- sozd.dat - файл с исходными данными для создания бинарного файла.
- poisk1.dat – файл для тестирования функции поиска по одному поисковому признаку (в каждой строке файла по одному поисковому данному);
- poisk2.dat - файл для тестирования функции поиска по сочетанию двух поисковых признаков (в каждой строке файла по два поисковых признака);
- dop.dat – файл, аналогичный по внешнему виду файлу sozd.dat с данными для дополнения бинарного файла;
- kor.dat – файл с данными для модификации записей бинарного файла (в каждой строке файла два данных: первое - для поиска структуры в бинарном файле и второе – новое значение заданного элемента найденной структуры).

2. Написать программу, в которой определить в соответствии с заданием функции обработки данных.
3. В главной функции произвести вызов функций. Вывод результирующей текстовой информации следует производить в текстовой файл результатов.

Пример выполнения лабораторной работы

Даны сведения о студентах: номер зачетной книжки, наименование группы, фамилия и инициалы, размер стипендии. Исходные данные для создания бинарного файла поместить в файл данных в виде, представленном на рис. 4.1 (файл sozd.dat).

Данные для дополнения бинарного файла разместить в текстовом файле в виде, показанном на рис. 4.2 (файл dop.dat).

Данные для тестирования функций поиска по фамилии студента, поиска по сочетанию двух признаков и функции коррекции записей в

бинарном файле показаны на рис. 4.3, 4.4, 4.5 (файлы poisk1.dat , poisk2.dat, kor.dat).

Листинг программы и текст файла результатов представлены на рис. 4.6. и 4.7 соответственно.

```
11007   ЭВМ 2-1   Иванов А.А.       280.5
11000   ЭВМ 2-1   Петров П.А.       167.9
11001   ЭВМ 2-2   Митин П.Ю.        0
11002   ЭВМ 2-2   Качнов А.И.       250.5
11004   ЭВМ 2-3   Васильева И.Б.    180.7
11005   ЭВМ 2-3   Горнец Н.Н.       167.9
```

Рис. 4.1. Содержимое файла данных sozd.dat

```
11012   ЭВМ 2-1   Козинов А.И.      240.0
11014   ЭВМ 2-2   Арапова Н.А.      0
11015   ЭВМ 2-3   Васина Р.П.       167.9
```

Рис. 4.2. Содержимое файла данных dop.dat

```
Митин П.Ю.

Качнов А.И.
Сидоров А.О.
```

Рис. 4.3. Содержимое файла данных poisk1.dat

```
250.5   ЭВМ 2-2
0       ЭВМ 2-1
200
120     ЭВМ 1-1
150     ЭВМ 2-3
```

Рис. 4.4. Содержимое файла данных poisk2.dat

```
Васильева И.Б.   250.5
                 0
Горнец Н.Н.      0
```

Рис. 4.5. Содержимое файла данных kor.dat

```
//Лабораторная работа №6 студента группы ЭВМ 1-1 Иванова Петра
#include <stdlib.h>
#include <string.h>
#include <iomanip.h>
#include <fstream.h>
#include <stdio.h>
struct stud{  long nz; //объявляется структурный тип stud и структура st
              char gr[8], fio [16];
              float rs;} st;
```

Рис. 4.6. Листинг программ

```
int z = sizeof(stud);
char *sh[] = {
 "                  СВЕДЕНИЯ О СТУДЕНТАХ                              ",
 "                                                                    ",
 "┌──────────┬──────┬──────────┬──────────┐                           ",
 "│НОМЕР      │ГРУППА│ФАМИЛИЯ   │РАЗМЕР    │                           ",
 "│ЗАЧЕТКИ    │      │ИНИЦИАЛЫ  │СТИПЕНДИИ │                           ",
 "├──────────┼──────┼──────────┼──────────┤                           ",
 "│          │      │          │          │                           ",
 "└──────────┴──────┴──────────┴──────────┘                           "};
ifstream fin;                   // определение входного файлового потока
ofstream fout;                  //выходного файлового потока
fstream io;                     //двунаправленного
void p( ), cht( ), sozd( ), dop( ), zf( ), poisk1( ), poisk2( ), kor(), filtr(char*),
psh( );                         //прототипы вызываемых функций:
void main ( )                   //главная функция
{fout.open("L6.res" );                  // открытие файла результатов
if (!fout) {cout<<"Ошибка при открытии файла результатов"; exit(0);}
sozd();         //вызовы функций
cht();
dop();
cht();
poisk1();
poisk2();
kor();
cht();
fout.close();//закрытие файла результатов
}
//---------------------------------------------------------------------------
void psh( )   // вывод строк шапки таблицы в файл результатов
{for(int i = 0;i < 5; i++)
fout << sh[i] << endl;}
//---------------------------------------------------------------------------
void p( )       // вывод одной строки таблицы в файл результатов
{fout << '\272' << setiosflags(ios::left) << setw(13) << st.nz << '\272'
 << setw(10) << st.gr << '\272' << setw(15) << st.fio << '\272'
 << setw(14) << setprecision(2) << st.rs << '\272' << endl;
}
//--чтение бинарного файла, вывод структур в файл результатов в таблицу:
void cht( ){int n = 0;                  //счетчик структур в бинарном файле
float s= 0;             // суммарная стипендия
fout << "\n            ЧТЕНИЕ ФАЙЛА\n";
fin.open ("binary.cpp", ios:: in|ios::binary); //открытие файла для чтения
if (!fin) {cout<<"Ошибка открытия бинарного файла для чтения"; exit(0);}
psh();// вывод строк шапки таблицы
```

Рис. 4.6. (Продолжение)

```
while (fin.peek( ) != EOF)        //пока не достигли символа конца файла
{fin.read( (char*) &st, z);                  //считываем структуру в переменную st
p( );                    //выводим поля структуры в таблицу
s += st.rs ; n++;}
fout << sh[5] << endl << "Суммарная стипендия = " << s << endl
 << "Средняя стипендия = " << s/n << endl
 << "Количество структур в файле = " << n << endl;
fin.close( );}
//-------------------------------------------------------------------------
void sozd( )          //-функция создания бинарного файла
{fout<< "\n          СОЗДАНИЕ  ФАЙЛА ";
fin.open("sozd.dat");                //открываем файл с исходными данными
if (!fin) {cout<<"Ошибка при открытии файла данных"; exit(0);}
//создаем новый бинарный файл для записи:
io.open ("binary.cpp", ios :: out|ios :: binary);
if (!io){cout<<"Ошибка открытия бинарного файла для создания"; exit(0);}
psh( );                  //вывод строк шапки таблицы в файл результатов
zf( ); //вызов функции чтения данных и записи их в бинарный файл:
fin.close( ); io.close( );}
//-----чтение данных из файла данных и запись данных в бинарный файл----
void zf( ) {char T[80];
while (!fin.eof( ) )
{ fin>>st.nz; fin.getline(st.gr,11); filtr(st.gr);fin.getline(st.fio,17); filtr(st.fio);
fin >> st.rs; fin.getline(T,80);
io.write((char*) &st, z);  // запись структуры целиком из ОП  в бин. файл
p( ); }          //запись полей структуры в строку таблицы файла результатов
fout<<sh[5]<<endl;}
//--------------определение функции filtr взять из ЛР №5-------------------
void filtr (char *s) {…}
//-------------------------------------------------------------------------------
void poisk1( )                //поиск данных по фамилии
{char name[20];    //символьный массив для хранения поискового признака
fin.open("poisk1.dat" );          //открытие файла данных для поиска
if (!fin) {cout<<"Ошибка открытия файла данных для поиска 1"; exit(0);}
fout<<"\n\n          ПОИСК ПО ФАМИЛИИ СТУДЕНТА\n";
io.open ("binary.cpp", ios :: in | ios :: binary);//открытие бин. ф. для чтения
if (!io) {cout<<"Ошибка открытия бинарного файла для чтения "; exit(0);}
while(fin.peek( )!=EOF){fin.getline(name,20); filtr(name);
fout << "\nИщем студента с фамилией -" << "\" "<< name <<" \"" << endl;
if(!strcmp(name,"")) {fout << "Нет фамилии для поиска\n"; continue;}
io.seekg (0,ios :: beg);     //указатель бин. файла устанавливается на начало
```

Рис.4.6. (Продолжение)

```
while(io.peek( )!=EOF)  //пока не достигнут символ конца файла
{io.read((char*)&st,z);    //читаем из файла одну структуру в st
if(strcmp(st.fio, name) == 0) //сравниваем поле fio структуры st с name
{ p( ); goto m; }}//если совпадение, выводим данные структуры
//и управление передается на метку m
fout<< "Студент с фамилией\" "<< name<<" \" не найден"<<endl;
m:;}
fin.close( ); io.close( );}
//--------------------поиск по сочетанию признаков------------------------------
void poisk2( ){char grup[15]; float stip;
//переменные для хранения поисковых признаков-
//это наименование группы и размер стипендии
fin.open("poisk2.dat" ,ios :: in);          // открытие файла данных для поиска
if(!fin) {cout<<"Ошибка открытия файла данных для поиска 2"; exit(0);}
fout << "\n\n ПОИСК ПО НАИМЕНОВАНИЮ ГРУППЫ СТУДЕНТА И"
"ПО РАЗМЕРУ СТИПЕНДИИ\n";
while (fin.peek() != EOF)          //пока не достигнут символ конца файла
{fin >> stip; fin.getline(grup, 15); filtr(grup); //считываются поиск. признаки
int k = 0;
fout << "\nИщем в группе -" << "\"" << grup << "\""
 <<" студентов со стипендией > = " << stip << endl;
if ((!strcmp(grup, "" )) &&(stip == 0.0))
{fout << "Нет для поиска ни группы, ни стипендии \n"; continue;}
io.open ("binary.cpp", ios :: in | ios :: binary); //открытие бин. ф. для чтения
if (!io) {cout<<"Ошибка открытия бинарного файла для чтения";exit(0);}
while(io.read((char*)&st,z))      //пока читаются данные из бинарного файла
if ((!strcmp(st.gr, grup) || !strcmp(grup, "")) && (stip <= st.rs || stip == 0))
{ p( ); k++; }//если условие истинно, поля выводятся
//и счетчик выводимых структур увеличивается на 1
if (!k) fout << "Таких студентов нет" << endl;
io.close( );} //конец тела цикла по файлу данных с поисковыми признаками
fin.close( ); io.close( );}
//---------------дополнение бинарного файла-----------------------------------
void dop( ){
fout<< "\n\n ДОПОЛНЕНИЕ ФАЙЛА НОВЫМИ ЗАПИСЯМИ" << endl;
fin.open("dop.dat", ios :: in); // открытие файла данных для дополнения
if(!fin) {cout<<"Ошибка открытия файла данных для дополнения"; exit(0);}
//открытие бинарного файла в режиме дополнения
io.open ("binary.cpp", ios :: app | ios :: binary);
if (!io){cout<<"Ошибка открытия бин. файла для дополнения"; exit(0);}
for(int i = 1;i < 5; i++)
fout<<sh[i]<<endl;
```

Рис. 4.6. (Продолжение)

```
zf( );
fin.close( ); io.close( );}
//--------------функция коррекции структур бинарного файла---------------------
void kor()
{char t[80];          //массив - буфер
char name[85]; //для хранения фамилии студента – поисковый признак
float stip;                    //переменная для новой стипендии
fout << "\n\n                  КОРРЕКЦИЯ ФАЙЛА\n\n";
fin.open("kor.dat", ios : :in);
if (!fin) {cout<<"Ошибка открытия файла данных для коррекции"; exit(0);}
while(fin.peek( ) != EOF)        //1 цикл ввода данных для коррекции
{fin.getline(name, 17); filtr(name); fin >> stip; fin.getline(t, 80);
int k=0;
if(!strcmp(name, ""))
{fout << "\nНет фамилии для поиска структуры\n\n"; continue;}
//открытие бинарного файла в режиме чтения и записи
io.open ("binary.cpp", ios :: in | ios :: out | ios :: binary);
if (!io){cout<<"Ошибка открытия бин. файла для его коррекции"); exit(0);}
while(io.read((char*)&st, z)) // 2
if(!strcmp(st.fio, name))              //если найдена структура
{p(); st.rs = stip; p();
io.seekp(-z, ios :: cur); //указатель файла на одну структуру назад
io.write((char*)&st, z); //запись в файл откорректированной структуры
k++ ; break;} // while 2
if (!k) fout<<"Студент с ФИО - " << name << " не найден" << endl;
io.close( );}          // while 1
fin.close( );io.close( );}
```

Рис. 4.6. (Продолжение)

СОЗДАНИЕ ФАЙЛА
СВЕДЕНИЯ О СТУДЕНТАХ

НОМЕР ЗАЧЕТКИ	ГРУППА	ФАМИЛИЯ ИНИЦИАЛЫ	РАЗМЕР СТИПЕНДИИ
11007	ЭВМ 2-1	Иванов И. И.	280.5
11000	ЭВМ 2-1	Петров Т.О.	167.9
11001	ЭВМ 2-2	Митин П.Ю.	0
11002	ЭВМ 2-2	Качнов А.И.	250.5
11004	ЭВМ 2-3	Горнец Н.Н.	180.7
11005	ЭВМ 2-3	Васильева И.Б.	167.9

Рис. 4.7. Содержимое файла результатов L6.res

ДОПОЛНЕНИЕ ФАЙЛА ЗАПИСЯМИ

НОМЕР ЗАЧЕТКИ	ГРУППА	ФАМИЛИЯ ИНИЦИАЛЫ	РАЗМЕР СТИПЕНДИИ
11012	ЭВМ 2-1	Козинов А.И.	240
11014	ЭВМ 2-2	Арапова Н.А.	0
11015	ЭВМ 2-3	Васина Р.П.	167.9

ЧТЕНИЕ ФАЙЛА
СВЕДЕНИЯ О СТУДЕНТАХ

НОМЕР ЗАЧЕТКИ	ГРУППА	ФАМИЛИЯ ИНИЦИАЛЫ	РАЗМЕР СТИПЕНДИИ
11007	ЭВМ 2-1	Иванов И. И.	280.5
11000	ЭВМ 2-1	Петров Т.О.	167.9
11001	ЭВМ 2-2	Митин П.Ю.	0
11002	ЭВМ 2-2	Качнов А.И.	250.5
11004	ЭВМ 2-3	Горнец Н.Н.	180.7
11005	ЭВМ 2-3	Васильева И.Б.	167.9
11012	ЭВМ 2-1	Козинов А.И.	240
11014	ЭВМ 2-2	Арапова Н.А.	0
11015	ЭВМ 2-3	Васина Р.П.	167.9

```
Суммарная стипендия = 1.46e+03
Средняя стипендия = 161.71
Количество структур в файле = 9
```

ПОИСК ПО ФАМИЛИИ СТУДЕНТА

```
Ищем данные студента с фамилией -" Митин П.Ю. "
||11001        ||ЭВМ 2-2   ||Митин П.Ю.     ||0              ||

Ищем данные студента с фамилией -"  "
Нет  фамилии для поиска

Ищем данные студента с фамилией -" Качнов А.И. "
||11002        ||ЭВМ 2-2   ||Качнов А.И.    ||250.5          ||

Ищем данные студента с фамилией -" Сидоров И.В. "
Студент с фамилией   " Сидоров И.В. " не найден
```

Рис. 4.7. (Продолжение)

```
 ПОИСК ПО НАИМЕНОВАНИЮ ГРУППЫ СТУДЕНТА И ПО РАЗМЕРУ СТИПЕНДИИ

Ищем в группе -"ЭВМ 2-2"студентов со стипендией > = 250.5
||11002         ||ЭВМ 2-2    ||Качнов А.И.     ||250.5          ||

Ищем в группе -"ЭВМ 2-1"студентов со стипендией > = 0
||11007         ||ЭВМ 2-1    ||Иванов И. И.    ||280.5          ||
||11000         ||ЭВМ 2-1    ||Петров Т.О.     ||167.9          ||
||11012         ||ЭВМ 2-1    ||Козинов А.И.    ||240            ||

Ищем в группе -""студентов со стипендией > = 200
||11007         ||ЭВМ 2-1    ||Иванов И. И.    ||280.5          ||
||11002         ||ЭВМ 2-2    ||Качнов А.И.     ||250.5          ||
||11012         ||ЭВМ 2-1    ||Козинов А.И.    ||240            ||

Ищем в группе -"ЭВМ 1-1"студентов со стипендией > = 120
Таких студентов нет

Ищем в группе -"ЭВМ 2-3"студентов со стипендией > = 150
||11004         ||ЭВМ 2-3    ||Горнец Н.Н.     ||180.7          ||
||11005         ||ЭВМ 2-3    ||Васильева И.Б.  ||167.9          ||
||11015         ||ЭВМ 2-3    ||Васина Р.П.     ||167.9          ||

                    КОРРЕКЦИЯ ФАЙЛА

||11005         ||ЭВМ 2-3    ||Васильева И.Б.  ||167.9          ||
||11005         ||ЭВМ 2-3    ||Васильева И.Б.  ||250.5          ||

Нет фамилии для корректировки

||11004         ||ЭВМ 2-3    ||Горнец Н.Н.     ||180.7          ||
||11004         ||ЭВМ 2-3    ||Горнец Н.Н.     ||0              ||

                ЧТЕНИЕ ФАЙЛА
                СВЕДЕНИЯ О СТУДЕНТАХ
```

НОМЕР ЗАЧЕТКИ	ГРУППА	ФАМИЛИЯ ИНИЦИАЛЫ	РАЗМЕР СТИПЕНДИИ
11007	ЭВМ 2-1	Иванов И. И.	280.5
11000	ЭВМ 2-1	Петров Т.О.	167.9
11001	ЭВМ 2-2	Митин П.Ю.	0
11002	ЭВМ 2-2	Качнов А.И.	250.5
11004	ЭВМ 2-3	Горнец Н.Н.	0
11005	ЭВМ 2-3	Васильева И.Б.	250.5
11012	ЭВМ 2-1	Козинов А.И.	240
11014	ЭВМ 2-2	Арапова Н.А.	0
11015	ЭВМ 2-3	Васина Р.П.	167.9

```
Суммарная стипендия = 1.36e+03
Средняя стипендмя = 150.81
Количество структур в файле = 9
```

Рис. 4.7. (Продолжение)

Контрольные вопросы

1. Что такое текстовые и бинарные файлы?
2. Как объявить текстовой и бинарный файлы?
3. Какие типы данных можно хранить в бинарном файле?
4. Какая функция используется для связи логического файла программы (файлового потока) с физическим файлом?
5. Средства обмена данными с потоками: операции ввода (>>) и вывода (<<) данных.
6. Средства обмена данными с потоками: функции двоичного ввода/вывода данных.
7. Какие функции используются для обмена данными между ОП и бинарным файлом?
8. Поясните назначение, параметры и возвращаемое значение для следующих функций: peek(), putback(), ignore( ), seekg( ), seekp(), tellg(), tellp(), close(), remove(), rename().
9. Поясните процесс обработки данных, схемы алгоритмов и тексты функций программы вашей лабораторной работы.

ЛИТЕРАТУРА

1. Подбельский В.В. Стандартный Си++. - М.: Финансы и статистика, 2008.
2. Климова Л.М. Основы практического программирования на языке С++. - М.: Приор, 2002.

СОДЕРЖАНИЕ