Лабораторная работа № 6

**Редактирование файлов: удаление, изменение поля**

**Цель:** Научиться создавать структуры с возможностью дальнейшего редактирования полей и удаления записей, изучить структурные шаблоны, структурные переменные.

**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

Тип переменной определяет её размер в памяти, тип данных, которые она может хранить и операции, которые можно производить с этой переменной.

Тип данных является категорией. В языке С (С++) программист может **создать любой тип данных на основе базовых типов**. Новые типы данных необходимо создавать для решения конкретных практических задач. Например – реализация работы со студенческой группой.

С помощью структур возможно моделировать сложные объекты, возникающие при решении задач. Структуры представляют средство для доступа к записям, которые содержат поля одного или нескольких типов.

Для использования структуры необходимо:

1. *объявить шаблона структуры* как нового типа данных. На этом этапе память не выделяется. Формируется только информация о содержимом структуры;

2. *объявить переменную*, соответствующую этому шаблону. На этом этапе выделяется память для любого поля (переменной), которое описывается в шаблоне структуры;

3. *осуществить доступ* к компонентам структуры.

***Структура*** представляет собой коллекцию объединенных ***общим именем*** переменных различного типа, у которых есть некоторая логическая взаимосвязь. Структуры иногда называют *составными* или *конгломератными типами данных.*

Создание структуры (или объединения) означает создание ***нового*** определенного программистом ***типа данных***.

## **Объявление структуры**

Прежде чем будет создан объект структуры, должен быть определен ее формат. Это делается посредством объявления структуры –установка структурного шаблона ***struct***. Объявление структуры позволяет понять, переменные какого типа она содержит. Переменные, составляющие структуру, называются ее членами или элементами, или полями. В общем случае все члены структуры должны быть логически связаны друг с другом.

***Член структуры*** *–* это переменная, которая является частью структуры.

Для объявления структуры используется ***структурный шаблон***, который является основной схемой, описывающей содержание структуры.

Общий формат объявления:

*struct имя\_ типа\_ структуры {*

*тип имя\_элемента 1;*

*тип имя\_элемента 2;*

*тип имя элемента 3;*

*…*

*тип имя\_элемента М;*

*}* ;

Например, объявим структуру студент, содержащую элементы – фамилия студента и его средний балл, название группы, например, uir1. Для этот создадим шаблон:

*struct stud*

*{*

*char name[20];*

*char group\_name[20];*

*float bal;*

*}****;***

Первым в шаблоне стоит ключевое слово ***struct***. Оно определяет, что все, что стоит за ним, является структурой. Далее следует необязательный «тег» (*имя типа структуры*) – слово *stud*, являющееся сокращенной меткой, которую можно использовать для ссылки на эту структуру. *Имя структуры* – это ее ***спецификатор типа***.

*Компоненты структуры (список элементов структуры)* заключены в парные фигурные скобки. Каждый элемент определяется своим собственным описанием. Например, элемент *name* является символьным массивом, состоящим из 20 элементов. *Элементы могут быть данными любого типа, включая другие структуры*.

Например, объявим структуру студенческой группы ***group***, содержащую элементы – уникальный идентификатор группы *id*, название группы *group\_name* и массив объектов типа ***stud***, который должен содержать данные о студентах группы. Создадим шаблон:

*struct stud*

*{*

*int id\_group;*

*char name[20];*

*float bal;*

*};*

***struct group***

*{*

*int id\_group;*

*char group\_name[20];*

***stud st[30];***

*}****;***

Все данные в структуре (*struct*) пишутся в фигурных скобках, и в конце ставится запятая с точкой (**;**), завершающая определение шаблона.

***!*** Любая структура должна начинаться с ключевого слова – ***struct***, которое сообщает компилятору, что будет создана структура. Объявление структуры представляет собой ***инструкцию***, поэтому оно заканчивается точкой с запятой (;).

Как видно из примера в *структурах* ***stud*** и ***group*** находятся данные различных типов, но они объединены в логическую связь. Данные в структуре должны иметь уникальные имена, но в различных структурах можно использовать одинаковые названия.

Можно разместить шаблон за пределами любой функции (вне), или внутри определения функции. Если установили шаблон внутри функции, то он может использоваться только внутри этой функции. Если вне, то шаблон доступен всем функциям программы, следующим за его определением. Установка шаблона не вызывает никаких действий в программе. Память для него не выделяется.

## **Создание объекта**

Слово «структура» используется двояко. Во-первых, в смысле «структурного шаблона». Шаблон является схемой без содержания; он сообщает компилятору, как делать что-либо, но не вызывает никаких действий в программе. Так при объявлении структуры *stud* в действительности не было создано ни одной переменной. Был определен лишь формат данных. Таким образом, структура, которая создана выше, не занимает в памяти компьютера места, так как, на самом деле, просто создали ***пользовательский тип данных***, аналогичный int, float.

Следующий шаг заключается в создании «структурной переменной»; это и есть второй смысл слова структура. Объявление структурных переменных приводит к выделению памяти для компонент структуры, куда можно записать данные или откуда можно прочитать их. Чтобы с помощью созданного структурного шаблона объявить реальную переменную (т.е. физический объект), можно воспользоваться несколькими способами.

**I способ:** Нужно записать инструкцию, объявляющую простую переменную, массив структур, указатель на структуру:

*void main(){*

*struct stud my, mas[5], \*pmy;*

*}*

Объявление ***структуры*** ничем не отличается от объявления любого типа данных в языке С (С++). На основании этого оператора компилятор создает структурную переменную типа *struct stud* с именем ***my***, массив ***mas***, состоящий из 5 элементов, типа *struct stud* и указатель *\*pmy* на переменную *struct stud*.

Таким образом, при определении структуры определяется новый тип данных, но он не будет реализован до тех пор, пока не будет объявлена переменная того типа, который уже реально существует.

При объявлении структурной переменной С (С++) автоматически выделит объем памяти, достаточный для хранения всех членов структуры. В примере, переменная my будет размещена в памяти компьютера (в предположении, что name – символьный массив 20 элементов – 20 байт, char-значение *group\_name* – символьный массив 20 элементов – 20 байт, а float-значение *bal* – 4).

Одновременно с определением структуры можно сразу после объявления структуры объявить одну или несколько переменных, как показано в следующем примере.

Общий формат объявления структуры в этом случае будет выглядеть так.

*struct имя\_ типа\_ структуры {*

*тип имя\_элемента1;*

*тип имя\_элемента2;*

*тип имя элемента 3;*

*тип имя\_элементаМ;*

*}* ***структурные\_переменные****;*

Объявление переменных данного типа:

*struct stud {*

*char name[20];*

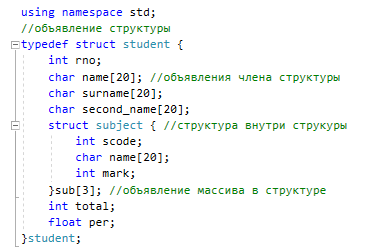
*char group\_name[20];*

*float bal;*

*}* ***my, mas[5], \*pmy;***

Этот фрагмент кода определяет структурный тип *stud* и объявляет переменные ***my, mas[5]*** и указатель ***\*pmy*** этого типа. Важно понимать, что каждая структурная переменная содержит собственные копии членов структуры. Например, поле *name* структуры *my* изолировано от поля *name* структуры *mas[5]*. Следовательно, изменения, вносимые в одно поле, никак не влияют на содержимое другого поля.

Пример:



Форма с именем типа структуры удобнее, если используется структурный шаблон более одного раза.

Можно ввести новый тип данных (***stud***)-структура определенного вида с помощью инструкции ***typedef.***

typedef struct stud {

char name[20];

char group\_name[20];

float bal;

}**;**

Объявить переменные нового типа:

***stud*** my, mas[5], \*pmy;

Если программа достаточно объемна, представляется более удобным данный способ.

## **Инструкция typedef**

Язык С (C++) позволяет определять имена новых типов данных с помощью ключевого слова ***typedef***. На самом деле здесь не создается новый тип данных, а определяется новое имя существующему типу. Объявления ***typedef*** можно использовать для создания более коротких и содержательных имен стандартным типам данных, определенных в языке, или для типов, которые объявили пользователи. Пример:

typedef int integer;

где int – это тип, которому присваиваем новое имя, а integer – это то обозначение, которое будет использоваться вместо int.

Имена *typedef* разделяют пространство имен с обычными идентификаторами. Поэтому программа может иметь имя typedef и идентификатор локальной области с одним и тем же именем.

## **Операции со структурами**

Операции со структурами:

1. присваивание полю структуры значение того же типа;

2. можно получить адрес структуры. Операция взятия адреса (&);

3. можно обращаться к любому полю структуры;

4. для того, что бы определить размер структуры можно использовать операцию sizeof().

Инициализировать можно только внешние или статические структуры. Здесь следует иметь в виду, что принадлежность структурной переменной к внешнему типу зависит от того, где определена сама переменная, а не где определен шаблон. В примере 1 на странице 5 шаблон *stud* является внешним, а переменная ***my*** (*struct stud my;*) – внутренней, так как она определена внутри функции *main()* и по умолчанию располагается в классе автоматической памяти.

**Инициализация структуры в языке** С (С++) происходит так же, как и при инициализации массива. Например:

struct stud my = { "Иванов", "УИР1", 7.5 };

Здесь создаем переменную типа *struct stud* и присваиваем всем трем полям, которые определенны в структуре, значения. Порядок очень важен при *инициализации структуры*, так как компьютер сам не может отсортировывать данные. Если какое-либо поле не будет заполнено, то оно автоматом заполнится 0 – для целочисленных типов; NULL – для указателей; \0 (ноль–терминатор) – для строковых типов. Компилятору требуются только запятые, чтобы отделить инициализацию одного элемента от инициализации следующего.

***Пример:*** Структура студенческой группы:

#include <iostream>

struct stud

{

char name[20];

float bal;

}**;**

**static struct**

**{**

**int id\_group;**

**char group\_name[20];**

**stud st;**

**}** **group[3] =**

**{**

**1, "группа 1", "Герасимчик Е.А.", 9.25 ,**

**2, "группа 2", "Романченко В.С.", 8.5,**

**3, "группа 3", "Шапкина Е.А.", 9.85,**

**};**

void main()

{

setlocale(LC\_ALL, "Russian");

std::cout << std::endl;

for (int i = 0; i < 3; i++)

{

std::cout << group[i].id\_group << '\t';

std::cout << group[i].group\_name << '\t';

std::cout << group[i].st.name << '\t' << '\t';

std::cout << group[i].st.bal << std::endl << std::endl;

}

}

При вводе данные для структуры **stud st;** можно было выделить с помощью скобок {}:

1, "группа 1", { "Герасимчик Е.А.", 9.25 }.

**Доступ к элементам структуры**

Обращаться в языке С (С++) к элементам структуры можно двумя способами:

**I способ:** через операцию – «точку(**.**)» – обозначающий операцию получения элемента структуры. Обычная точка выполняет операцию доступа к элементам.

Общий формат доступа:

***имя\_структурной\_переменной* · *имя\_члена***

Оператор «точку(**.**)» позволяет получить доступ к любому члену любой структуры.

***Пример***: Создать структуру stud как в предыдущих примерах, инициализировать ее и вывести на экран полученные данные.

#include <iostream>

struct stud {

char name[20];

char group\_name[20];

float bal;

};

void main()

{

setlocale(LC\_ALL, "Russian");

struct stud my = { "Герасимчик Е.А.", "УИР 2 Ср.балл = ", 9.25 };

printf("\n %s %s %f\n\n", my.name, my.group\_name, my.bal);

// вывод значения элемента name

// и bal структуры stud

my.bal = 2; //присваиваем элементу bal новое значение

}

В примере, *my.bal* является элементом bal структурной переменной ***my*** структуры *stud*. В сущности, ***.name***, ***.group\_name*** и ***.bal*** являются индексами для структуры *stud*.

**II способ:** через операцию – «стрелку (–>)».

Обозначение: «**–>»**. Различия заключаются в том, что операцию - точку используем, когда работаем с переменной, а операцию - стрелку – когда с указателем на переменную. Пример:

#include <iostream>

#include <stdio.h>

struct stud {

char name[20];

char group\_name[20];

float bal;

};

void main()

{

setlocale(LC\_ALL, "Russian");

struct stud my = { "Герасимчик Е.А.", "УИР 2 Ср.балл = ", 9.25 };

**struct stud \*pmy = &my;**

printf("\n %s %s %f\n\n", pmy**->**name, pmy**->**group\_name, pmy**->**bal);

printf("\t\t\t Ср.балл = %.1f\n\n", pmy**->**bal);

}

Обязательно надо ставить знак взятия адреса (**&**) перед именем структуры, так как имя структуры, в отличии от имени массива, не является указателем. Можно обратиться к элементам структуры через указатель другим способом:

printf("%.1f", (\*pmy).bal);

При таком обращении надо брать в скобки указатель (\*ptr), т.к. операция доступа имеет более высокий приоритет по сравнению с операцией разыменования (\*).

**Структура** является **типом данных** созданным пользователем, к ней можно применять такие же операции, как и к встроенным типам. Рассмотрим работу структур (**struct**) с функциями.

ПРИМЕР:

//объявление структуры

typedef struct student {

int rno;

char name[20]; //объявления члена структуры

char surname[20];

char second\_name[20];

struct subject { //структура внутри струкуры

int scode;

char name[20];

int mark;

}sub[3]; //объявление массива в структуре

int total;

float per;

}student;

ФУНКЦИЯ ЗАПОЛНЕНИЯ

void create() {

student\* s;

FILE\* fp;

int n, i, j;

char cifra;

printf("Введите количество студентов : ");

scanf("%d", &n);

s = (student\*)calloc(n, sizeof(student));

fp = fopen("mystudents1.txt", "w+");

for (i = 0; i < n; i++) {

s[i].total = 0;

s[i].per = 0;

printf("Введите номер студента : ");

scanf("%d", &s[i].rno);

fflush(stdin);

printf("Введите имя : ");

scanf("%s", s[i].name);

printf("Введите фамилию : ");

scanf("%s", s[i].surname);

printf("Введите отчество : ");

scanf("%s", s[i].second\_name);

for (j = 0; j < 3; j++) {

printf("Введите оценки студента%d : ", j + 1);

scanf("%d", &s[i].sub[j].mark);

s[i].total += s[i].sub[j].mark;

}

s[i].per = s[i].total / 3.0;

//cifra = s[i].per;

fwrite(s + i, sizeof(student), 1, fp);

}

fclose(fp);

}

ФУНКЦИЯ ИЗМЕНЕНИЯ:

void update() {

int i, j, found = 0;

student s1;

FILE\* fp, \* fp1;

int rno;

printf("Введите номер студента для обновления данных : ");

scanf("%d", &rno);

fp = fopen("mystudents1.txt", "r");

fp1 = fopen("temp.txt", "w");

while (fread(&s1, sizeof(student), 1, fp)) {

if (s1.rno == rno) {

found = 1;

s1.total = 0;

s1.per = 0;

//fflush(stdin);

printf("Введите новое имя : ");

scanf("%s", s1.name);

printf("Введите новую фамилию : ");

scanf("%s", s1.surname);

printf("Введите новое отчество : ");

scanf("%s", s1.second\_name);

for (j = 0; j < 3; j++) {

printf("Введите новые оценки студента%d : ", j + 1);

scanf("%d", &s1.sub[j].mark);

s1.total += s1.sub[j].mark;

}

s1.per = s1.total / 3.0;

}

fwrite(&s1, sizeof(student), 1, fp1);

}

fclose(fp);

fclose(fp1);

if (found) {

fp = fopen("mystudents1.txt", "w");

fp1 = fopen("temp.txt", "r");

while (fread(&s1, sizeof(student), 1, fp1)) {

fwrite(&s1, sizeof(student), 1, fp);

}

fclose(fp);

fclose(fp1);

}

else

printf("\nНе найдено.\n");

}

ФУНКЦИЯ УДАЛЕНИЯ

void delete\_rec() {

int i, j, found = 0;

student s1;

FILE\* fp, \* fp1;

int rno;

printf("Введите код студента для удаления : ");

scanf("%d", &rno);

fp = fopen("mystudents1.txt", "r");

fp1 = fopen("temp.txt", "w");

while (fread(&s1, sizeof(student), 1, fp)) {

if (s1.rno == rno) {

found = 1;

}

else

fwrite(&s1, sizeof(student), 1, fp1);

}

fclose(fp);

fclose(fp1);

if (found) {

fp = fopen("mystudents1.txt", "w");

fp1 = fopen("temp.txt", "r");

while (fread(&s1, sizeof(student), 1, fp1)) {

fwrite(&s1, sizeof(student), 1, fp);

}

fclose(fp);

fclose(fp1);

}

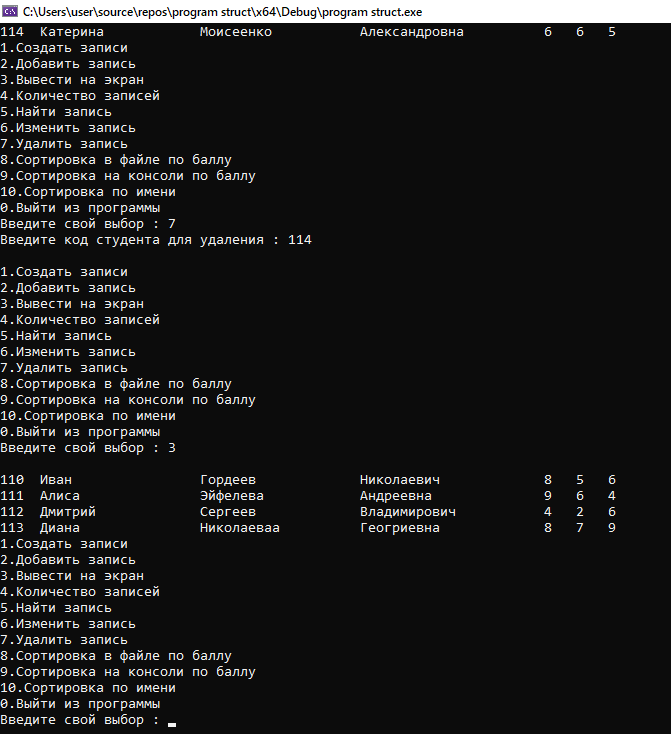
else

printf("\nНе найдено\n");

}

РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ:





# ЗАДАНИЯ:

К имеющемся меню из лабораторной работы №1 добавить функции:

* редактировать
* удалить

Описать структуру с именем STUDENT, содержащую следующие поля:

• NAME – фамилия и инициалы;

• GROUP – номер группы;

• SES – оценки по пяти предметам (массив из пяти элементов).

Написать программу, реализующую следующие действия отдельными функциями:

• ввод с клавиатуры данных в массив STUD, состоящий из N переменных типа STUDENT;

• вывод на экран и запись в файл фамилий и номеров групп для всех студентов, оценки.

• изменение количества студентов, сохранение изменений в файле.

• удаление записей в файле.