Лабораторная работа №8

Структуры.

Цель работы: освоить принципы работы со структурами в Си, выполнить упражнения по вариантам

Краткие теоретические сведения

Языки программирования C/C++ поддерживает определяемые пользователем структуры – структурированный тип данных. Он является *собранием одного или более объектов (переменных, массивов, указателей, других структур и т.д.), которые для удобства работы с ними сгруппированы под одним именем*.

Структуры:

- облегчают написание и понимание программ.
- помогают сгруппировать данные, объединяемые каким-либо общим понятием.
- позволяют группу связанных между собой переменных использовать как множество отдельных элементов, а также как единое целое.

Как и массив, структура представляет собой *совокупность данных*, но *отличается* от него тем, что *к ее элементам (компонентам) необходимо обращаться по имени* и ее элементы могут быть *различного типа*. Структуры целесообразно использовать там, где необходимо объединить данные, относящиеся к одному объекту.

Определение структуры состоит из двух шагов:

- объявление шаблона структуры (задание нового типа данных, определенного пользователем);
- определение переменных типа объявленного шаблона.

Объявление шаблонов структур. Общий синтаксис объявления ***шаблона структуры***:

```
struct имя_шаблона
{
    тип1 имя_переменной1;
    тип1 имя_переменной1;
    //другие члены данных;
};
```

```
struct DateBase
{
    char fam[20];
    char name[15];
    long TelNumber;
    char *Adress;
    double w;
};
```

Имена *шаблонов* должны быть *уникальными* в пределах их *области определения* для того, чтобы компилятор мог различать различные типы шаблонов. Задание шаблона осуществляется с помощью ключевого слова **struct**, за которым следует имя шаблона структуры и список элементов, заключенных в фигурные скобки.

Имена элементов *в одном шаблоне* также должны быть *уникальными*. Однако в разных шаблонах можно использовать одинаковые имена элементов.

Задание только шаблона *не влечет* резервирования памяти компилятором. Шаблон представляет компилятору необходимую информацию об элементах структурной переменной для *резервирования места* в оперативной памяти и организации доступа к ней *при определении* структурной переменой и использовании отдельных элементов структурной переменной.

Среди членов данных структуры могут также присутствовать, кроме стандартных типов данных (**int**, **float**, **char** и т.д.), ранее определенные типы, например:

```
/* объявление шаблона структуры типа date */
        struct date
        { int day, month, year; };
/* шаблон структуры person */
        struct person
        {
            char fam[30], im[20], otch[15];
            float weight;
            int height;
            struct date birthday;
        };
```

Структура *date* имеет три поля типа **int**. Шаблон структуры *person* в качестве элемента включает поле *birthday*, которое, в свою очередь, имеет ранее объявленный тип данных: **struct** *date*. Этот элемент (*birthday*) содержит в себе все компоненты шаблона **struct** *date*.

Определение структур-переменных. Определение структуры-переменной ничем не отличается от объявления обычной переменной с предопределенным типом. Общий синтаксис:

```
struct имя_шаблона имя_переменной;
```

Пример:

```
        struct person stud[10];
```

Компилятор выделит под каждую переменную количество байтов памяти, необходимое для хранения всех ее элементов.

Разрешается совмещать описание шаблона и определение структурной переменной.

Пример:

```
  struct book
  {
            char title[20];
            char autor[30];
            double cast;
  } book1, book2, *ptr_bk=&book1;
```

Пример:

```
  struct date {int day, month, year;} date1[5]; // объявление массив из 15 структур
```

Доступ к компонентам структуры. Доступ к полям осуществляется с помощью оператора «**.**» при непосредственной работе со структурой или «->» - при использовании указателей на структуру. Эти операторы называются *селекторами* членов класса. Общий синтаксис для доступа к компонентам структуры следующий:

```
имя_переменной_структуры.член_данных;
имя_указателя->имя_поля;
(*имя_указателя).имя_поля;
```

***Пример**:*
Прямой доступ к элементам

1) date1[5].day=10;
2) date1[5].year=1991;
3) **strcpy**(book1.title, “Война и мир”);

/* используя прямое обращение к элементу, присваиваем значение выбранной переменной. Текс помещается в переменную, используя функцию копирования – strcpy(); */

4) stud[3].birthday.month=1;
5) stud[3].birthday.year=1980;

Доступ по указателю

1) (date1+5)->day=10;
2) (stud+3)->birthday.month=1;

// Используя доступ по указателю на структуру, присваиваем значение соответствующей переменной. Указатель можно использовать и так:

3) (*(date1+5)).day=10;
4) (*(stud+3)).birthday.month=1;

Инициализация структур. При определении структурных переменных можно инициализировать их поля. Эта возможность подобна инициализации массива и следует тем же правилами:

имя_шаблона имя_переменной_структуры = {значение1, значение2, …};

Компилятор присваивает *значение1* первой переменной в структуре, *значение2* – второй переменной структуры и т.д., и тут необходимо следовать некоторым правилам:

- присваиваемые значения должны совпадать по типу с соответствующими полями структуры;
- можно объявлять меньшее количество присваиваемых значений, чем количество полей. Компилятор присвоит нули остальными полями структуры;
- список инициализации последовательно присваивает значения полям структуры, вложенных структур и массивов.

Пример:

```
struct date
{ int day,month,year; }d[5] = { {1,3,1980}, {5,1,1990}, {1,1,1983} };
```

Копирование структур-переменных. Язык С(С++) позволяет оператору *присваивания* копировать значения одной структуры-переменной в другую переменную, при условии, что обе структуры-переменные относятся к одному и тому же типу. Таким образом, единственный оператор может скопировать несколько членов данных, которые включают массивы и вложенные структуры.

Однако следует учитывать, что оператор присваивания выполняет то, что называется *поверхностной* копией в применении к структурам-переменным. Поверхностная копия представляет собой копирование бит за битом значений полей переменной-источника в соответствующие поля переменной-цели. При этом может возникнуть проблема с такими членами данных, как *указатели*, поэтому использовать поверхностное копирование структур надо осторожно.

Пример 1: Создать массив структур о студентах группы. О каждом студенте записать: имя, фамилию, год рождения, оценки по пяти экзаменам. Определить средний балл за сессию и отсортировать список по сумме баллов.

```
#include "stdafx.h"
#include<stdio.h>
#include<conio.h>

struct student
{ char name[20];
  char fam[30];
  int year;
  int mark[5];
  int average;
};

struct student students[30];
struct student buffer;
int records;
int i, j;
int main()
{     records=0;
      do
      {
            printf("Student №%d\n", records+1);
            puts("Vvedite familiu: ");
            fflush(stdin);
            gets(students[records].fam);
            puts("Vvedite imya: ");
            fflush(stdin);
            gets(students[records].name);
            puts("Vvedite vozrast: ");
            scanf("%d", &students[records].year);
            for(i=0; i<5; i++)
            {     printf("Vvedite ocenku po ekzamenu №%d ", i+1);
                  scanf("%d", &students[records].mark[i]);
            }
            records++;
            puts("Prekratit' rabotu? (1/0)");
            scanf("%d", &i);
      } while(i);

      for (i=0; i<records; i++)
      { students[i].average=0;
        for (j=0; j<5; j++)
              students[i].average+=students[i].mark[j];
      }

      for (i=0; i<records-1; i++)
            for (j=i; j<records; j++)
                  if (students[i].average>students[j].average)
                  {  buffer=students[i];
                     students[i]=students[j];
                     students[j]=buffer;
                  }

      for (i=0; i<records; i++)
      {
            printf("Student %s %s ", students[i].name, students[i].fam);
            printf(" Vosrast %d ", students[i].year);
            printf(" Sr. ball %d \n", students[i].average);
      }
      getch();
      return 0;
    }
```

Пример 2: Ввести массив структур. Рассортировать массив в алфавитном порядке фамилий, входящих в структуру, перемещая сами структуры.

```
#include "stdafx.h"
#include<stdio.h>
#include<conio.h>
#include<string.h>

struct st
{ char name[80];
  int age; не придирайся
};

int _tmain(int argc, _TCHAR* argv[])
{
      int i,j,k;
      struct st m[100], t;
      printf("Vvedite kol-vo studentov: ");
      scanf("%d", &k);
      for (i=0; i<k; i++)
      {     puts("Vvedite imya studenta: ");
            fflush(stdin);
            gets(m[i].name);
            puts("Vvedite vozrast: ");
            scanf("%d", &m[i].age);
      }
      for (i=0; i<k-1; i++)
            for (j=i+1; j<k; j++)
                  if (strcmp(m[i].name, m[j].name)>0)
                  {  t=m[i]; m[i]=m[j]; m[j]=t; }
      printf("\n\nRezultat: ");
      for (i=0; i<k; i++)
      {
            printf("\n\n");
            puts(m[i].name);
            printf("%d years ", m[i].age);
      }
      getch();
      return 0;
    }
```

Практическая часть

Ввести массив структур в соответствии с вариантом. Рассортировать массив в алфавитном порядке по первому полю, входящему в структуру. В программе реализовать меню:

1) Ввод массива структур;
2) Сортировка массива структур;
3) Поиск в массиве структур по заданному параметру;
4) Изменение заданной структуры;
5) Удаление структуры из массива;
6) Вывод на экран массива структур;
7) Выход.

Варианты:

1. Структура «Автосервис»: регистрационный номер автомобиля, марка, пробег, мастер, выполнивший ремонт, сумма ремонта.
2. Структура «Сотрудник»: фамилия, имя, отчество; должность; год рождения; заработная плата.
3. Структура «Человек»: фамилия, имя, отчество; домашний адрес; номер телефона; возраст.
4. Структура «Читатель»: Фамилия И.О., номер читательского билета, название книги, срок возврата.
5. Структура «Студент»: фамилия, имя, отчество; домашний адрес; группа; рейтинг.
6. Структура «Покупатель»: фамилия, имя, отчество; домашний адрес; номер телефона; номер кредитной карточки.
7. Структура «Клиент банка»: Фамилия И.О., номер счета, сумма на счете, дата последнего изменения.
8. Структура «Авиарейсы»: номер рейса, пункт назначения, время вылета, дата вылета, стоимость билета.
9. Структура «Вокзал»: номер поезда, пункт назначения, дни следования, время прибытия, время стоянки.
10. Структура «Кинотеатр»: название кинофильма, сеанс, стоимость билета, количество зрителей.