СТРУКТУРЫ в Си – ПРОГРАММАХ. СЕМАНТИКА И СИНТЕЗ

1. ***Способы описания структуры и определение ее элементов.***
   Выделение памяти под структуры. Примеры.

2. ***Доступ к элементам структуры. Адресные выражения.***
   Примеры.

3. ***Примеры программ , использующих структуры.***

4. ***Массивы структур.***

5. ***Указатели на структуру.***

6. ***Структуры и функции.***

1. Способы описания (объявления) структуры. Примеры.

- Описание всякой структуры в программе начинается с ключевого слова ***struct*** и в простейшем случае имеет следующий формат:

```
struct {member-declaration list}
        identifier <,identifier ... >;
```

- ***struct*** есть ключевое слово языка Си, а в угловые скобки (**<>**) заключена необязательная часть конструкции.

- ***member-declaration list*** - одно или более описаний переменных, каждая из которых называется элементом структуры, а ***identifier*** - имя переменной, определяемой как имеющей тип структура (называется экземпляром структуры).

Когда объявлявляется структура, то для нее компилятор выделяет память, в которой могут размещатся элементы (поля) разных типов

```
// ключевое слово struct

STRUCT идентификатор {перечень полей_структуры};

STRUCT  A  // идентификатор - имя
 {
   char B[5];
   int count;
};
```

поля

Знак <;> обязателен !

ниже объявлена более сложная структура:

Еще пример *(объявление сложной структуры)*

```
STRUCT A
{  int count;
   char b[20];
};

STRUCT B
{ int out;
  A c;
};

struct B f[2];
```

//объявлена структура A
//тогда после нее пусть дано такое описание:
***struct A d;* //объявление переменной *d,* которая содержит**
//в себе две величины: символьный массив с именем *b,* //переменную целого типа с именем *count*

//в описании объявлена переменная f – массив структур,
//содержащих в качестве поля другую структуру,
//имеющую поле - массив

Ниже показаны другие способы объявления:

```
//структурный тип задает служебное слово typedef

typedef struct                    //структурный тип не именован
{
  int real;
  int imag;
} complex; // complex – экземпляр (переменная) структурного_типа
```

два поля структуры

```
//структурный тип не именован
struct
 {
   int a;
   float b;
   char c;
 } sample1, sample2;
```

экземпляры структурного типа

2. Доступ к элементам структуры. Адресные выражения. Примеры.

d={“может”, 7};//d-структура

cout << d.B<<“ ”<<d.count;//B, count-элементы

// т.е. доступ выполнен по правилу:

//имя_структуры . имя_элемента

Пример операций в операторе печати *printf()*:

```
…
char ch;
struct strel
{ int k;
  char * fc;
  char * st;
} a, deck[25], * cPtr;
```

Здесь указатели – полей структуры

Экземпляры структуры

```
…
printf ("%s", a . st );
…
cPtr = &a;
…
printf ("%s", cPtr → st);
```

Здесь использованы две операции доступа:
-операция “точка”;
-операция “стрелка”.

//Далее показан доступ (извлечение из структуры):

```
a.strel=&ch;
```

3. Примеры программ , использующих структуры.

```
*Пример программы: сложение комплексных чисел*/
include <stdio.h> // прототип функций ввода\выода (printf, puts, scanf и др.)
include <conio.h> // прототип функции getch ()
ypedef struct     // определение и объявление структуры  typedef- задание
                  // структурного типа
    {         //начало описания
      int real;    //список описания структуры
      int imag;
    }      //конец описания
    complex;    //complex – объявление переменной структурного типа
oid main ()        //объявление главной функции
                   //начало описания главной функции main()
complex c1, c2, c3, read ();         //переменные комплексного типа и функция
   void  add  (complex,  complex,  complex*),  print  (complex);        //объявление
рототипа
                                                          //  функций add и print;
c1 = read ();  //чтение с клавиатуры числовых данных
c2 = read();
add (c1, c2, &c3);
printf ("при сложении "); print (c1); printf (" и "); pri
printf ("\nполучилось "); print (c3);
getch ();    //задержка экрана
```

! Протестировать самостоятельно

```
complex read ()          //тип функции complex
 {//                                  ----------------------
   complex c;//                                                |
   puts ("введите действительную и мнимую части числа:"); /*
функция
          выводит текст */
  scanf ("%d%d", &(c.real), &(c.imag));//читает с клавиатуры
  return c; // и записывает введённые с клавиатуры данные в память
} //
                                  ----------------------
void print (complex c)      //функция печати комплексных чисел
 {
   printf ("%d+i*(%d)", c.real, c.imag);
 }

void add (complex c1, complex c2, complex*c3)   //функция подсчёта суммы
                                                 // комплексных чисел
 {
   c3-> real = c1.real + c2.real; //c3 присваивается сумма действит.  чисел
   c3-> imag = c1.imag + c2.imag; //затем сумма мнимых чисел
}
```

Пример записи данных в сложную структуру *mybox*

```
//простая структура типа coord для хранения координат точки:
struct coord {
        int x;
        int y;
      }
//сложная структура rectangle, содержащая две структуры, которые
//задают противоположные углы прямоугольника,
//объявляется экземпляр структуры
struct rectangle {
       struct coord topleft;
       struct coord bottomrt;
     } mybox;
…
//помещение (запись) значений в поля mybox:
mybox.topleft.x = 100; mybox.bottomrt.x = 300;
mybox.bottomrt.y = 400; mybox.topleft.y = 200;
…
```

Пример фрагмента программы:

```
//Вычисление длины, ширины и площади
//прямоугольника по данным предыдущей структуры
...

width= mybox.bottomrt.x - mybox.topleft.x;
length = mybox.bottomrt.y - mybox.topleft.y;
...

// Вычисление площади прямоугольника
area = width * length;
...
```

4. Массивы структур.

```
// определение структуры с полями из массивов
struct A
 {
   char B[3];
   char C[5];
   char D[7];
 };

// объявление и печать массива структур в некоторой программе
…

struct A mas[4];

…
//пример печати:
for(i=0;i<4;i++) scanf(“%s”, mas[i].B);
…
```

/*Программа работы с массивами структур*/

```
#include <stdio.h>
#include <conio.h>
struct entery //структура
 {
  char fname [20];
  char lname [20];
  char phone [10];
};
struct entery list [4];//массив
int i;
void main ()
{
 clrscr();
 printf("\nПрограмма ввода/
    вывода данных 2-х
персон:");
 for(i=0;i<2;i++)
  {
   printf("\n\nведите
                первое имя->");
   scanf("%s", list[i]. fname);
// см. продолжение
```

```
//продолжение

  printf("\n введите второе имя -> ");
   scanf("%s", list[i].lname);
   printf("\n введите телефон в формате
            \"***-**-**\" -> ");
   scanf("%s", list[i].phone);
 } //конец оператора цикла

 printf("\n\n");
 for(i=0;i<2;i++)
  {
   printf(" Name : %s %s",
   list[i].fname, list[i].lname);
   printf(" \t\t Phone : %s\n",
   list[i].phone);
  }
  getch();
} // end main
```

5.Указатели на структуры и операции над ними. (ниже дано объявление и инициализация)

```
//объявим структуру
struct A
 {
   short count;
   char B[5];
};
//объявим указатель на значение типа A
struct A *p_A;

//перед инициализацией указателя объявим хотя бы
//один экземпляр типа A (что и сделано выше)

struct A giz;

//инициализируем указатель
p_A = &giz;
```

//*Пример:* перебор массива структур с помощью указателей

```
//начало
#include <iostream.h>
#define n 3
struct fio
{
   short num;
   char name[10];
}
data [n]={
1, “Ivan”, 2, “Petro”, 3, “Nikolai”
};
struct fio *p_fio;
int k;
int main (void)
{
  p_fio=data;
```

```
 //продолжение
 for (к=0;k<n;k++)
 {
   cout << “код:”<<p_fio<<
                    p_fio->num;
   p_fio++;
 }
} //end main
```

6. Структуры и функции
(изучить самостоятельно !)

- Совершенно очевидно, что отдельные элементы структур, являющиеся простыми переменными или указателями произвольного типа, могут быть использованы в качестве аргументов при обращении к функциям.

- Однако более важным является вопрос о возможности передачи через аппарат формальных/фактических параметров структур в целом. Эту операцию наиболее естественно осуществить, используя понятие указателя на структуру.

- Для иллюстрации технических деталей, связанных с передачей и обработкой структур, рассмотрим фрагмент программы, отыскивающей в сводном каталоге книгу, имеющую наиболее ранний год издания. Общая организация данных, необходимая для решения этой задачи, может быть представлена при помощи структурного шаблона **BOOK.**

```
//Пример программы с именем BOOK
#include <stdio.h>
#define MAX 300

struct BOOK { char author[30]; // Автор книги
              char title[256]; // Название книги
              int year;        // Год издания
              int pages;       // Количество страниц
             };

/* Поиск самой старой книги */

int find(book) struct BOOK *book;
{ int cnt, min;
   min = book->year;
   for (cnt = 0; cnt < MAX; cnt++, book++)
       if (book->year < min) min = book->year;
   return (min);
}
```

```
//продолжение
void main()
 { int min_year;
   struct BOOK catalog[MAX];
   ...
   min_year = find(catalog);
   printf("\nСамая старая книга издана в %d году", min_year);
}
```

!!! **Примечание.** Некоторые реализации языка Си допускают использование структур как единого целого в качестве аргументов функций, передавая по значению отдельные элементы таких структур.