Урок 9. Структуры и объединения

В языках C/C++ понятие структуры - это структурированный тип данных, представляющий собой поименованную совокупность разнотипных элементов. Тип структура обычно используется при разработке информационных систем, баз данных.

Правила использования структур обсудим на примере. Сведения о выплате студентам стипендии требуется организовать в виде, показанном на рис. 1.

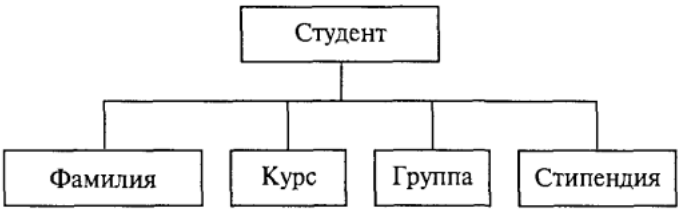


Рис. 1.

Элементы такой структуры (фамилия, курс, группа, стипендия) называются полями. Каждому полю должно быть поставлено в соответствие имя и тип.
Формат описания структурного типа следующий:

```
struct имя_типа
{определения_элементов};
```

В конце обязательно ставится точка с запятой (это оператор). Для рассмотренного примера определение соответствующего структурного типа может быть следующим:

```
struct  student
{
char  fam[30];
int  kurs;
char  grup[3];
float  stip;
};
```

После этого student становится именем структурного типа, который может быть назначен некоторым переменным. В соответствие со стандартом C это нужно делать так:

```
struct student studl, stud2;
```

Правила C++ разрешают в этом случае служебное слово struct опускать и писать

```
student studl, stud2;
```

Здесь studl и stud2 — переменные структурного типа.

Допускаются и другие варианты описания структурных переменных. Можно вообще не задавать имя типа, а описывать сразу переменные:

```
struct
{
char  fam[30];
int   kurs;
char  grup[3];
float stip;
} studl, stud2, *pst;
```

В этом примере кроме двух переменных структурного типа объявлен указатель pst на такую структуру. В данном описании можно было сохранить имя структурного типа student.

Обращение к элементам (полям) структурной величины производится с помощью уточненного имени следующего формата:

```
имя_структуры.имя_элемента
```

Примеры уточненных имен для описанных выше переменных:

```
studl.fam;  studl.stip
```

Значения элементов структуры могут определяться вводом, присваиванием, инициализацией. Пример инициализации в описании:

```
student studl = {"Кротов", 3, "Ф32", 350};
```

Пусть в программе определен указатель на структуру student *pst, studl;
Тогда после выполнения оператора присваивания

```
pst = &studl;
```

к каждому элементу структурной переменной studl можно обращаться тремя способами. Например, для поля fam studl.fam или (*pst).fam или pst->fam.
В последнем варианте используется знак операции доступа к элементу структуры: ->.
Аналогично можно обращаться и к другим элементам этой переменной:

```
pst->FIO, pst->grup, pst->stip.
```

Поля структуры могут сами иметь структурный тип. Такие величины представляют многоуровневые деревья.

Допускается использование массивов структур. Например, сведения о 100 студентах могут храниться в массиве, описанном следующим образом:

```
student stud[100];
```

Тогда сведения об отдельных студентах будут обозначаться, например, так:

```
studfi].fam,  stud[5].kurs
```
и т.п.

Если нужно взять первую букву фамилии 25-го студента, то следует писать:

```
stud[25].fam[0].
```

Пример 1. Ввести сведения об N студентах. Определить фамилии студентов, получающих самую высокую стипендию.

```
#include  <stdio.h>
#include  <conio.h>
void  main( )
{
const N = 30; int i;
float maxs;
struct student
{
char  fam[15];
int kurs;
char grup[3];
float stip;
};
```

```
student  stud[N];
void  clrscr();
for(i = 0; i < N; i++)
{
printf("%d-n  студент", i);
printf("\nФамилия:");
scanf("%s", &stud[i].fam);
printf("Курс:");
scanf("%d", &stud[i].kurs) ;
printf("Группа:");
scanf("%s",&stud[i].grup);
printf("Стипендия:");
scanf("%f",&stud[i].stip);
}
maxs = 0;
for(i=0; i<N; i++)
if(stud[i].stip>maxs)  maxs = stud[i].stip;
printf("\n Студенты, получающие максимальную стипендию %f руб.", maxs);
for(i = 0; i < N; i++)
if(stud[i].stip == maxs) printf("\n%s", stud[i].fam);
}
```

Элемент структуры типа поля битов. Использование структуры в программе на С позволяет работать с отдельными битами, т.е. с разрядами двоичного кода. Для этого используются элементы структуры типа поля битов. Формат структуры, содержащий поля битов, следующий:

```
struct имя_структуры
{
тип имя_поля 1: длина_в_битах;
тип имя_поля 2: длина в битах;
тип имя_поля_3: длина_в_битах;
};
```

В качестве типа полей могут использоваться спецификаторы int, unsigned, signed. Минимальной величиной такого типа может быть структура, состоящая всего из одного битового поля.
Пример описания такой структуры:

```
struct onebit
{
unsigned bit:1;
} cod;
```

Конечно, для переменной cod в памяти будет выделено 8 бит (1 байт), но использоваться будет только один первый бит.

Объединение. Объединение — это еще один структурированный тип данных. Объединение похоже на структуру и в своем описании отличается от структуры тем, что вместо ключевого слова struct используется слово union.

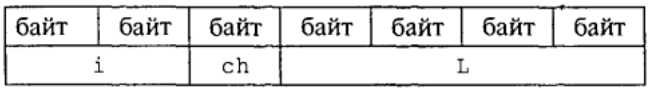
```
union имя_типа
{определения_элементов};
```

Объединение отличается от структуры способом организации во внутренней памяти. Все элементы объединения в памяти начинаются с одного байта.

Пусть в программе описана структура:

```
struct  S
{
int i;
char ch;
long int L;
};
```

Расположение ее элементов в памяти будет следующим:

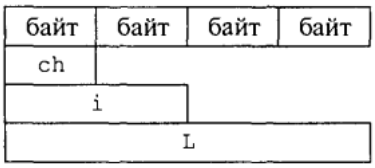


Элементы структуры занимают последовательные ячейки памяти с размером, соответствующим типу. Общий размер структуры равен сумме длин полей.

А теперь рассмотрим объединение со следующим описанием:

```
union s
{
int i;
char ch;
long int L;
};
```

Величина с таким типом в памяти будет расположена следующим образом:



Поля объединения накладываются друг на друга. Общий объем занимаемой памяти равен размеру самого большого поля. Изменение значения любого поля объединения меняет значения других полей.

Пример 2. Составим программу решения следующей задачи: с клавиатуры вводится символ. Вывести на экран двоичный код этого символа.

```
//Получение двоичного кода символа
#include <stdio.h>
#include <conio.h>
//Структура битовых полей
struct byte{
int   b1:1;
int   b2:1;
int   b3:1;
int   b4:1;
int   b5:1;
int   b6:1;
int   b7:1;
int   b8:1;
};
//Объединение - переменная
union bits
{
```

```
char ch;
byte cod;
} u;
//Прототип функции  декодирования
void decode(bits x);
//Основная  программа
void main( )
{
do{u.ch=getche();  //Ввод символа с клавиатуры
printf(":") ;
decode(u);
} while(u.ch! = 'q'); //Символ q-признак конца ввода
//Функция  декодирования
}
void decode(bits u)
{
if(u.cod.b8) printf("l"); else printf ("0")
if(u.cod.b7) printf("l"); else printf ("0")
if(u.cod.b6) printf("l"); else printf ("0")
if(u.cod.b5) printf("l"); else printf ("0")
if(u.cod.b4) printf("l"); else printf ("0")
if(u.cod.b3) printf("l"); else printf ("0")
if(u.cod.b2) printf("l"); else printf ("0")
if(u.cod.b1) printf("l"); else printf ("0")
printf("\n");
}
```

В этой программе переменная-объединение и содержит два элемента: символьное поле ch и битовую структуру cod, которые накладываются друг на друга. Таким образом, оказывается возможным получить доступ к каждому биту кода символа. Работа программы заканчивается после ввода символа q.

Вот вариант результатов работы данной программы:

```
s:01110011
d:01100100
J:01101010
a:01100001
b:01100010
c:01100011
d:01100100
q:01110001
```

Упражнения

1. Сведения о каждом химическом элементе из периодической таблицы Д. И. Менделеева представить в виде структуры. Написать программу ввода таблицы в память компьютера.
2. Представить координаты точки в трехмерном пространстве в виде структуры, состоящей из трех вещественных полей. Написать программу ввода координат двух точек и вычисления расстояния между ними.
3. Рассматривая комплексное число как структуру, состоящую из двух вещественных полей, составить функции выполнения четырех арифметических операций с комплексными числами.

Информационные источники

1. Семакин И.Г., Шестаков А.П. Основы программирования: Учебник.-М.: Мастерство, 2002.-432 с.