ЛЕКЦИЯ 6

Перечисления
Структуры
Объединения
Битовые поля

Перечисления

Перечисление – тип данных производный от целого типа, представляющий собой набор мнемонических констант.

Объявление перечисления имеет следующий синтаксис:

enum [имя типа] **{**элемент №1,...,элемент №N**}**
[список переменных];

Синтаксис элемента:
идентификатор [= значение]

Перечисления

Идентификатор может быть любым не использовавшимся ранее. Принято, что идентификаторы в перечислениях содержат заглавные буквы латинского алфавита, цифры и знак подчеркивания и начинаются только с буквы.

Значение должно быть целочисленной константой, его можно не указывать. В последнем случае, элементу будет присвоено значение на единицу большее, чем предыдущему элементу. Если это первый элемент в перечислении, то значение ноль.

Пример объявления перечисления:

```
enum WEEK {MONDAY = 1, TUESDAY, WEDNESDAY,
           THURSDAY, FRIDAY, SATURDAY, SUNDAY}
    day = MONDAY;
```

Перечисления

При последующих объявлениях переменных перечислимого типа используется следующий синтаксис:

enum тип имя [= значение];

Указывать ключевое слово **enum** обязательно. Например:

enum WEEK newday = FRIDAY;

ПРИМЕЧАНИЕ: Если при объявлении перечисления имя типа не указывать, то создать переменные данного перечисления можно только непосредственно в объявлении.

Перечисления

Чтобы избежать необходимости каждый раз при объявлении переменных перечислимого типа указывать ключевое слово **enum**, нужно объявить перечисление используя оператор **typedef**. Синтаксис такого объявления имеет вид:

typedef enum {элементы} имя типа;

Например:

typedef enum {MONDAY = 1, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY**}** WEEK;

Пример

```
printf(“Введите номер дня недели:”);
scanf(“%d”,&day);
switch(day){
  case MONDAY:    {printf(“Понедельник\n”); break;}
  case TUESDAY:   {printf(“Вторник\n”); break;}
  case WEDNESDAY: {printf(“Среда\n”); break;}
  case THURSDAY:  {printf(“Четверг\n”); break;}
  case FRIDAY:    {printf(“Пятница\n”); break;}
  case SATURDAY:  {printf(“Суббота\n”); break;}
  case SUNDAY:    {printf(“Воскресенье\n”); break;}
}
```

Структуры

Структура – это сложный тип данных представляющий собой упорядоченное в памяти множество элементов различного типа. Каждый элемент в структуре имеет свое имя и называется **полем**. Элементы в структуре располагаются последовательно. Размер структуры определяется суммой размеров всех элементов.

Структуры

Объявление структуры имеет вид:

struct [имя типа] **{**
  поле №1;
  поле №2;
  ...
  поле №N;
} [список переменных];

Объявление полей структуры возможно только без инициализации. Если несколько полей следующих друг за другом в описании структуры имеют один и тот же тип, то для их описания можно использовать синтаксис объявления нескольких переменных одного и того же типа. Типом поля может быть любой тип (как системный, так и пользовательский), описанный ранее.

Примеры структур

Структура, содержащая информацию о точке в двумерном пространстве (координаты):

```
struct Point{
  double x, y;
};
```

Структура, содержащая информацию об окружности (координаты центра и радиус):

```
struct Circle{
  double x, y, radius;
};
```

Примеры структур

Структура, содержащая информацию о студенте (фамилия, имя, отчество, номер зачетной книжки, средний балл):

```
struct Student{
  char surname[15], name[15], patronymic[15];
  unsigned number;
  double rate;
};
```

Структура, содержащая информацию о группе студентов (название группы, количество студентов, список студентов (максимально 30)):

```
struct Group{
  char name[10];
  unsigned number;
  struct Student list[30];
};
```

Структуры

Объявление переменной определенной структуры осуществляется после описания данной структуры в следующем виде:

struct тип имя №1[= значение №1][,...];

Примеры объявлений

```
struct Point pnt[3] = {{0,0},{1,0},{0,1}};

struct Circle c1 = {10.0,10.0,5.0},
                  c2 = {0.0,0.0,25.0};

struct Student st
            = {“Иванов”,”Иван”,”Иванович”,959623,7.5};

struct Group gr ={
  “97-ВС”, 3, {
      {“Иванов”, ”Иван”, ”Иванович”, 979601,8.0},
      {“Петров”, ”Петр”, ”Петрович”, 979602,6.5},
      {“Сидоров”,”Сидор”,”Сидорович”,979603,9.0}
    }
};
```

Структуры

От обязательно использования ключевого слова **struct** можно отказаться, если описывать структуру, используя оператор объявления типа **typedef** в следующем виде:

typedef struct [первичное имя типа] **{...}** имя типа;

Первичное имя типа, указываемое перед перечнем полей структуры является необязательным и указывается редко. Как правило, первичное имя типа имеет тот же идентификатор, что и основное имя типа, но начинается со знака подчеркивания.

Структуры

Структура, содержащая информацию о книге (ФИО автора, название книги, год издания):

```
typedef struct {
  char author[20], title[50];
  unsigned year;
} BOOK;
```

Объявление переменной данного типа:

BOOK book = {“А. Дюма”,”Три мушкетера”, 1986};

Структуры

Обращение к полям структуры осуществляется в следующем виде:

имя_переменной.имя_поля

Сначала указывается имя переменной структуры, а затем, через точку, имя поля. С точки зрения языка С при таком обращении к полю его значение может выступать как LValue, так и RValue значения.

Примеры

Вычисление длины окружности, заданной переменной *cir* типа Circle:

```
double length = 2.0*3.1415*cir.radius;
```

Ввод информации о студенте в переменную *st* типа Student:

```
scanf("%s %s %s %u %lf", &st.surname, &st.name,
        &st.patronymic, &st.number, &st.rate);
```

Примеры

Вывод на экран списка группы, заданной в переменой gr типа Group:

```
printf("Группа: %s\n",gr.name);
for(unsigned i=0;i<gr.number;i++)
  printf("%2u: %15s %15s %15s %6u %.1lf\n",
         i+1, gr.list[i].surname, gr.list[i].name,
         gr.list[i].patronymic, gr.list[i].number,
         gr.list[i].rate);
```

Структуры

Для определения размера переменной структурного типа в байтах используется оператор определения типа **sizeof**. Например:

```
unsigned size = sizeof(struct Student);    //size == 57
```

Объединения

Объединение – это сложный тип данных представляющий собой множество элементов различного типа, хранящихся по одному адресу. Каждый элемент в объединении имеет свое имя и называется **полем**.

Элементы в объединении располагаются на одном и том же пространстве памяти, перекрывая друг друга. Размер объединения определяется размером самого большого по размеру элемента.

Объединения

Объявление объединения имеет вид:

union [имя типа] **{**

  поле №1;

  поле №2;

  ...

  поле №N;

} [список переменных];

Объединения

Объявление переменных объединения имеет тот же синтаксис, что и для объявления переменных структур. Отличие состоит в том, что инициализировать значение объединения можно только значением первого поля. Например:

```
union VALUE{
  unsigned num;
  double val;
  char str[20];
};
union VALUE val = {0};
```

Объединения

Так же как и со структурами, для объединения можно создать свой тип данных, используя оператор **typedef**:

typedef union [первичное имя типа] **{...}** имя типа;

Объединения

Обращение к полям объединения имеет тот же синтаксис, что и обращение к полям структуры. При обращении к полю объединения данная конструкция может рассматриваться и как LValue и как RValue.

Определение размера памяти, занимаемого значением типа объединение, осуществляется оператором **sizeof**. Например:

```
unsigned size = sizeof(union VALUE); //size == 20
```

Пример

Рассмотрим, структуру для хранения одного из значений (символ, незнаковый символ, короткое целое число, короткое целое незнаковое число, целое число, целое незнаковое число, длинное целое, длинное незнаковое целое, вещественное число одинарной точности, вещественное число двойной точности):

Пример

```
typedef enum {
  INT_8 = 0,
  UINT_8,
  INT_16,
  UINT_16,
  INT_32,
  UINT_32,
  INT_64,
  UINT_64,
  FLOAT_32,
  FLOAT_64,
  FLOAT_80
} TYPE;
```

```
typedef union{
  char int8;
  unsigned char uint8;
  short int16;
  unsigned short uint16;
  int int32;
  unsigned uint32;
  long long int64;
  unsigned long long uint64;
  float float32;
  double float64;
  long double float80;
} VALUE;

typedef struct {
  TYPE type;
  VALUE value;
} VARIANT;
```

Пример

Работа с переменной типа VARIANT осуществляется в два действия:

- обращение к полю type (тип TYPE) для установления типа хранимого значения;
- обращение к элементу объединения VALUE (поле value) установленного типа.

Например, запись в переменную *val* типа VARIANT целого числа 150 будет иметь вид:

```
val.type = INT_32;
val.value.int32 = 150;
```

Пример

Вывод на экран значение переменной val типа VARIANT:

```
switch(val.type){
  case INT_8: printf("%d",val.value.int8); break;
  case UINT_8: printf("%u",val.value.uint8); break;
  case INT_16: printf("%hd",val.value.int16); break;
  case UINT_16: printf("%hu",val.value.uint16); break;
  case INT_32: printf("%d",val.value.int32); break;
  case UINT_32: printf("%u",val.value.uint32); break;
  case INT_64: printf("%lld",val.value.int64); break;
  case UINT_64: printf("%llu",val.value.uint64); break;
  case FLOAT_32: printf("%f",val.value.float32); break;
  case FLOAT_64: printf("%lf",val.value.float64);break;
  case FLOAT_80: printf("%Lf",val.value.float80);break;
}
```

Битовые поля

Битовое поле – последовательность бит длиной до 32 бит. В языке С битовое поле может быть только элементом структуры или объединения.

С точки зрения значения битовое поле рассматривается как целочисленная величина соответствующего размера.

Битовые поля

Описание битового поля имеет вид:
[**unsigned** | **int**] имя:размер;

В зависимости от того, какой тип данных был указан (**int** или **unsigned**) битовое поле будет знаковым или незнаковым. Например, объявим следующую структуру с двумя битовыми полями:

```
typedef struct{
   int a:4;
   unsigned b:4;
} BITFIELD1;
```

```
typedef struct{
   int a:24;
   unsigned b:24;
} BITFIELD2;
```

Диапазон значений принимаемых полем *a* структуры BITFIELD1 – [-8,7], а поля *b* – [0,15].

Диапазон значений принимаемых полем *a* структуры BITFIELD2 – [-8388608, 8388607], а поля *b* – [0, 16777216].

Пример

Рассмотрим структуру содержащую информацию о человеке:

- фамилия, имя, отчество (строки по 15 символов);
- дата рождения (в формате дд.мм.гггг);
- пол (М | Ж);
- номер мобильного телефона (семизначное число);
- номер домашнего телефона (шестизначное число);
- почтовый индекс города (шестизначное число);
- названия города и улицы (строки по 15 символов);
- номера дома и квартиры (целые числа).

Пример

```
typedef struct{
  char surname[15],
       name[15],
       patronymic[15];
  char sex;
  unsigned char day, month;
  unsigned short year;
  unsigned char mobile[7],
                home_tel[6];
  unsigned short house,
                 flat;
  unsigned char index[6];
  char town[15], street[15];
} MAN1;
```

```
typedef struct{
  char surname[15],
       name[15],
       patronymic[15];
  unsigned sex:1;
  unsigned day:5, month:4;
  unsigned year:14;
  unsigned mobile:24,
           home_tel:20;
  unsigned house:10,
           flat:10;
  unsigned index:20;
  char town[15], street[15];
} MAN2;
```

Пример 1

Дан список групп студентов. Каждая группа характеризуется: название группы (строка 10 символов), количество студентов в группе (целое число), список студентов (максимально 30 элементов). Каждый элемент содержит следующую информацию о студенте: ФИО (строки по 15 символов), номер зачетной книжки (целое шестизначное число), средний балл студента (вещественное число). Вывести список групп студентов со списком студентов в каждой группе, упорядоченным по среднему баллу, и указанием среднего балла каждого студента. Информация о каждом студенте вводится и выводится построчно в формате: *ФИО [номер зачетной книжки] средний балл*.

Пример 1 (объявления)

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  typedef struct{
    char fio[3][16];
    unsigned number;
    double rate;
  } STUDENT;
  typedef struct{
    char name[11];
    unsigned num;
    STUDENT list[30];
  } GROUP;
  unsigned num = 0;
  printf("Введите число групп: "); scanf("%u",&num);
  GROUP groups[num];
```

Пример 1 (ввод)

```
for(unsigned i=0;i<num;i++){
    printf("Введите информацию о группе %u\n",i+1);
    printf("Имя: "); scanf("%s",&groups[i].name);
    printf("Число студентов: ");
    scanf("%u",&groups[i].num);
    printf("Введите список студентов: \n");
    for(unsigned j=0;j<groups[i].num;j++){
      printf("%2u: ",j+1); //Вывод номера в списке
      scanf("%s %s %s [%u] %lf",
            &groups[i].list[j].fio[0],
            &groups[i].list[j].fio[1],
            &groups[i].list[j].fio[2],
            &groups[i].list[j].number,
            &groups[i].list[j].rate);
    }
    puts("---------------------------------------");
  }
```

Пример 1 (обработка)

```
for(unsigned i=0;i<num;i++){
    int flag = 1;
    while(flag){
      flag = 0;
      for(unsigned j=0;j<groups[i].num-1;j++)
        if(groups[i].list[j].rate < groups[i].list[j+1].rate){
          STUDENT st = groups[i].list[j];
          groups[i].list[j] = groups[i].list[j+1];
          groups[i].list[j+1] = st;
          flag = 1;
        }
    }
  }
```

Пример 1 (вывод)

```
  puts("Результат:");
  for(unsigned i=0;i<num;i++){
    printf("Группа %s (%u студентов)\n",
          groups[i].name,groups[i].num);
    double mid = 0.0;
    for(unsigned j=0;j<groups[i].num;j++){
      printf("%2u: %15s %15s %15s [%6u] %.1lf\n",j+1,
        groups[i].list[j].fio[0], groups[i].list[j].fio[1],
        groups[i].list[j].fio[2], groups[i].list[j].number,
        groups[i].list[j].rate);
        mid += groups[i].list[j].rate;
    }
    mid /= groups[i].num;
    printf("Средний балл: %lf\n",mid);
    puts("----------------------------------------");
  }
  return 0;
}
```

Пример 2

Дан список учащихся школ и ВУЗов. Каждый элемент списка содержит следующую информацию: ФИО учащегося (строка 45 символов), пол (М | Ж), дата рождения (в формате дд.мм.гггг). Если это школьник, то содержится следующая информация: номер школы (целое число), номер класса (целое число), буква класса (символ). Если это студент, то содержится следующая информация: ВУЗ (строка 10 символов), группа (строка 10 символов). Вывести список учащихся по типам: сначала школьники, а затем студенты. Внутри списков учащихся упорядочивать по ФИО. Информация о школьниках вводится и выводится построчно в следующем формате:

ФИО пол дата рождения, школа класс.

Информация о студентах вводится и выводится построчно в следующем формате:

ФИО пол дата рождения, ВУЗ группа.

Пример 2 (объявления)

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
  typedef enum{PUPIL = 0, STUDENT} TYPE;
  typedef struct{
    unsigned char school, form;
    char letter;
  } P_INFO;
  typedef struct{
    char uni[11], group[11];
  } S_INFO;
  typedef struct{
    char fio[46];
    unsigned pol:1,dd:5,mm:4,yy:14;
    TYPE type;
    union {P_INFO p_info; S_INFO s_info;} data;
  } MAN;
  unsigned num = 0;
  printf("Введите количество записей: "); scanf("%u",&num);
  if(num < 2) {printf("Слишком мало элементов! \n"); return 0; }
  MAN list[num];
```

Пример 2 (ввод)

```
for(unsigned i=0;i<num;i++){
  char str[100],fio[3][16],pol;
  unsigned data[3];
  fflush(stdin); gets(str);
  sscanf(str,"%s %s %s %c\
    %u.%u.%u",
    &fio[0],&fio[1],&fio[2],
    &pol,&data[0],&data[1],
    &data[2]);
  list[i].dd = data[0];
  list[i].mm = data[1];
  list[i].yy = data[2];
  strcpy(list[i].fio,"");
  for(int j=0;j<3;j++){
    strcat(list[i].fio,fio[j]);
    strcat(list[i].fio," ");
  }
```

```
list[i].pol= (pol=='M')?0:1;
  char *ptr = strchr(str,',');
  int res = sscanf(ptr+1,"%u %u%c",
          &list[i].data.p_info.school,
          &list[i].data.p_info.form,
          &list[i].data.p_info.letter);
  if(res == 3) list[i].type = PUPIL;
  else{
    list[i].type = STUDENT;
    sscanf(ptr+1,"%s %s",
      &list[i].data.s_info.uni,
      &list[i].data.s_info.group);
  }
}
```

Пример 2 (обработка)

```
int flag = 1;
while(flag){
  flag = 0;
  for(unsigned i=0;i<num-1;i++)
    if((list[i].type>list[i+1].type)||
       ((list[i].type==list[i+1].type)&&
        (strcmp(list[i].fio,list[i+1].fio)>0))){
      MAN tmp = list[i];
      list[i] = list[i+1];
      list[i+1] = tmp;
      flag = 1;
   }
}
```

Пример 2 (вывод)

```
printf("Результат:\n");
 for(unsigned i=0;i<num;i++){
   printf("%-45s %c %2u.%2u.%4u, ",
     list[i].fio,
     ((list[i].pol)?'Ж':'М'),
     list[i].dd,list[i].mm,list[i].yy);
   if(list[i].type == PUPIL)
     printf("%d %d%c\n",
       list[i].data.p_info.school,
       list[i].data.p_info.form,
       list[i].data.p_info.letter);
   else
     printf("%s %s\n",
       list[i].data.s_info.uni,
       list[i].data.s_info.group);
 }
 return 0;
}
```