Модуль 9. Составные типы данных

Рассматриваются перечислимый тип данных, структуры, объединения и массивы структур

Перечисления (**enum**)

Формат:

```
enum [ имя_типа ] { список_констант };
```

Примеры:

1.

```
enum spectr {red, orange, yellow, green, blue, violet};
spectr b;
…
    b=orange;
```

2.

```
enum {two = 2, three, four, ten = 10, eleven,
fifty = 50};
```

Структуры (struct)

```
struct [ имя_типа ] {
тип_1 элемент_1;
тип_2 элемент_2;
...
тип_n элемент_n;
} [ список_имен_объектов-структур ];
```

```
struct {
   char fio[30];
   int age, code;
   float money;
}stud[100], *ps;
```

```
struct worker{    char
fio[30];
   int age, code;
   float money;
 };
worker stud[100], *ps,
       teacher[15];
```

Инициализация структур

```
struct{
   char fio[30];
   int age, code;
   float money;
}worker = {“Ivanov P.I.“,18,215,3400.55};
```

```
struct complex {
   float real, im;
} compl [2][3]={
{{1, 1}, {1, 1}, {1, 1}},
{{2, 2}, {2, 2}, {2, 2}}
};
```

Доступ к полям структуры

```
worker worker1, stud[100], *ps;

worker1.fio = “Petrov A.V.";
stud[8].code = 215;
ps->money = 200.12;



struct A {int a; double x;};
struct B {A a; double x;} y[2];
y[0].a.a = 1;
y[1].x = 0.1;
```

Пример работы с массивом структур (вывод полей)

```
void print_worker(Worker);   //объявление функции
int main()
{
worker stud[100];
... /* формирование массива stud */
for (int i = 0; i<100; i++)
  print_worker(stud[i]); /* вызов функции */
}
void print_worker(worker w)  //определение функции
{
      cout << w.fio << ' '
                << w.age << ' '
         << w.code << ' '
         << w.money;
}
```

Битовые поля

```
struct Options {
   bool centerX:1;
   bool centerY:1;
   unsigned int shadow:2;
   unsigned int palette:4;};
```

```
struct{
short a: 12;
short b: 10;
} Pr;
```

Не использовано	b – 10 бит	a- 12 битов

Всего 4 байта или 32 бита

```
struct
{
short a: 12;
short : 4;
short b: 10;
} Pr ;
```

Не использовано	b – 10 бит	4 бита	a – 12 битов

Объединения (union)

union un {int a; float b; char c[20];};
un x;

```
#include <iostream.h>
int main(){
enum paytype {CARD, CHECK};
    paytype ptype;
    union payment{
        char card[25];
        long check;
    } info;
    /* присваивание значений info и ptype */
    switch (ptype){
        case CARD: cout << "Оплата по карте: "                <<
info.card; break;
        case CHECK: cout << "Оплата чеком: "
                << info.check; break;
    }}
```

Реализация доступа к битам в байте

```
union
{
unsigned char c;
struct
{
unsigned b0: 1;
unsigned b1: 1;
unsigned b2: 1;
unsigned b3: 1;
unsigned b4: 1;
unsigned b5: 1;
unsigned b6: 1;
unsigned b7: 1;
} byte;
} cod;
…
cod.c = 'a';
printf("%u", cod.byte.b4);
```

Ограничения объединений

- объединение может инициализироваться только значением его первого элемента;
- объединение не может содержать битовые поля;
- объединение не может содержать виртуальные методы, конструкторы, деструкторы и операцию присваивания;
- объединение не может входить в иерархию классов.

Типы данных, определяемые пользователем

enum

struct

union

Переименование типов (**typedef**)

typedef тип новое_имя [размерность];

```
typedef unsigned int UINT;
typedef char Msg[100];
typedef struct{
   char fio[30];
   int age, code;
   float money;} Worker;
```

```
UINT i, j;   Msg str[10];
Worker stud[100];
```

Массив структур: описание и доступ

```
struct book {
char title[100]; //наименование книги
char author[100]; //автор
int year; //год издания
};
int main()
{
int cnt_book = 0, ch;
struct book lib[100];
do
{
printf("Введите наименование книги: ");
scanf("%s",lib[cnt_book].title);
printf("Введите автора книги: ");
scanf("%s",lib[cnt_book].author);
printf("Введите год издания книги: ");
scanf("%d",&lib[cnt_book].year);
printf("Нажмите 0 для завершения ввода: ");
cnt_book++;
}
while(scanf("%d",ch) != 0 && cnt_book < 100);
return 0;
}
```