Объединения

Понятие «объединения»

Объединение — тип данных, включающий в себя поля, которые при размещении в памяти имеют нулевое смещение (не «поле за полем», а «поле внутри поля»).

Объединение обеспечивает доступ к одному и тому же участку памяти с помощью переменных (массивов/структур) разных типов.

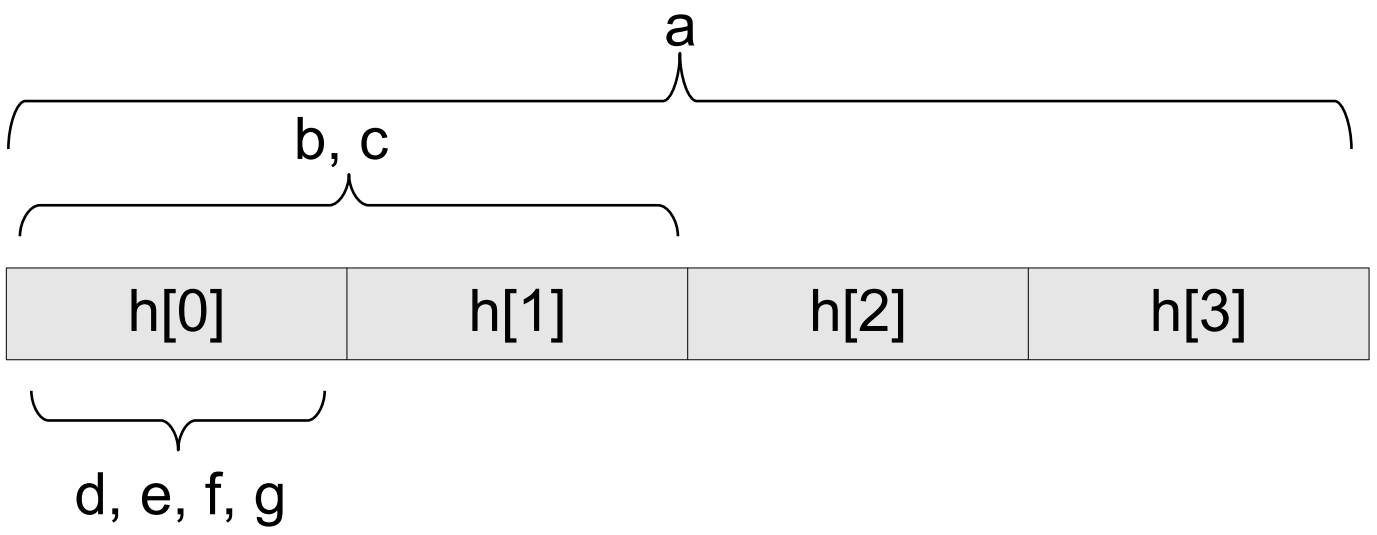
Назначение:

- Экономия памяти
- Выделение отдельных байтов из представления числа.

Для объявления объединения используется ключевое слово ***union***.

Размещение элементов объединения в памяти

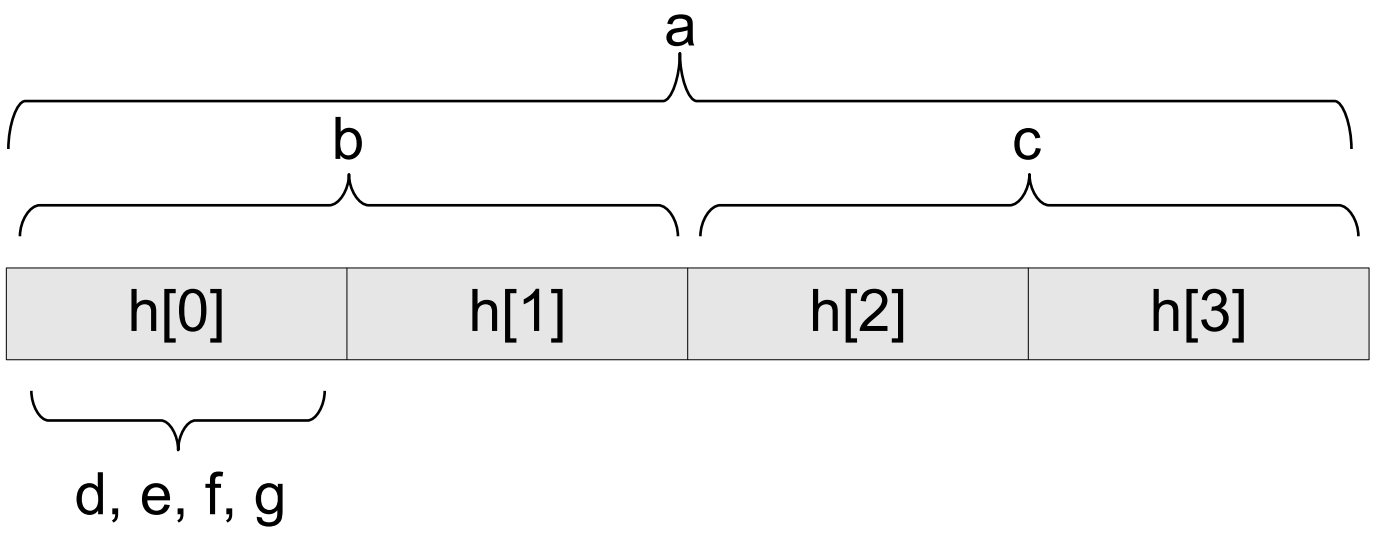
```
union
{
   int a;
   short b,c;
   char d,e,f,g;
   char h[4];
} un0; /* name of entity */
```



Все «укладывается» в область, выделенную для переменной a.

Размещение элементов объединения в памяти

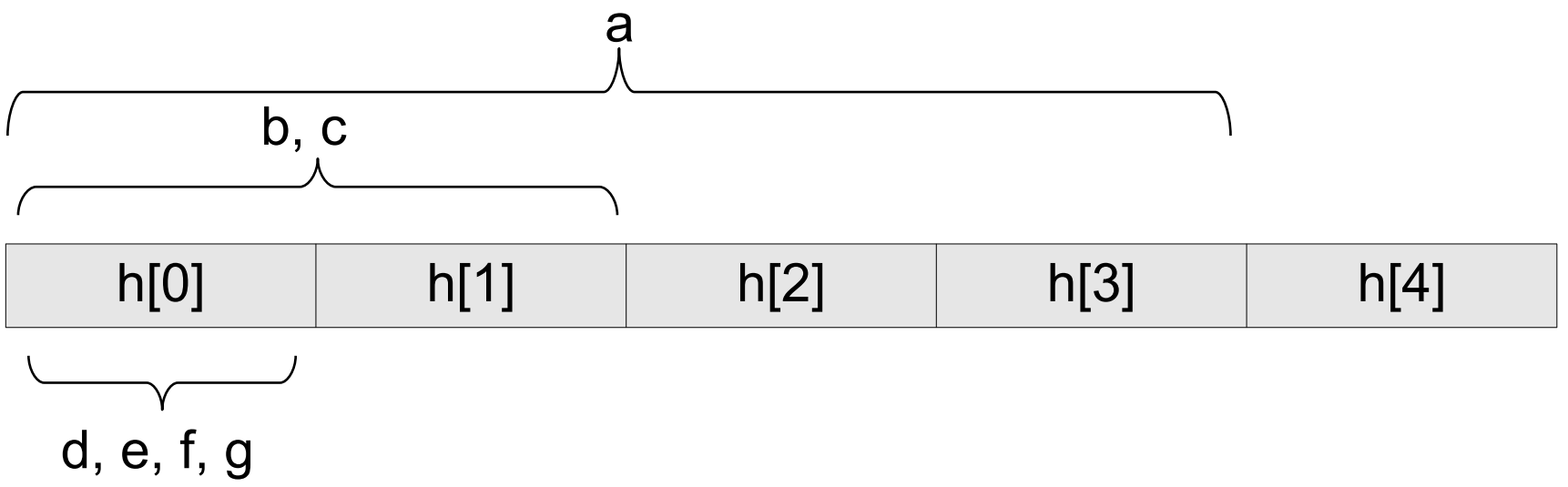
```
union
{
   int a;
   struct {
      short b;
      short c;
   } s0
   char d,e,f,g;
   char h[4];
} un0; /* name of entity */
```



Поля структуры располагаются «друг за другом» в пространстве адресов

Размещение элементов объединения в памяти

```
union
{
   int a;
   short b,c;
   char d,e,f,g;
   char h[5];
} un0; /* name of entity */
```



Все «укладывается» в область, выделенную для массива h.

Действует `#pragma pack()` !!

Обращение к полям объединения

```
union
{
   int a;
   short b,c;
   char d,e,f,g;
   char h[4];
} un0; /* name of entity */
```

Синтаксис для работы с полями объединения такой же, как и с полями структур:

```
scanf("%hd", &un0.b);

printf("%c",un0.h[2]);
```

См. пример lect-18-01.c, можно считывать элементы символьного массива, а можно — целые числа.

Особенности объединения

- Все элементы размещаются от начала одного и того же участка памяти
- Размеры участка памяти, выделяемого для объединения, определяются размером самого большого из элементов.
- Каждый элемент объединения использует фрагмент битового представления самого большого элемента.

```
typedef union
{
    int p1;
    float p2;
} T;
```

Если получить вещественное число как поле объединения, а потом применить к целочисленному полю битовый сдвиг, то получим двоичное представление вещественного числа (пример lect-18-02.c)

```
void fp(int z)
{
    int i;

    for(i=31;i>=0;i--) printf("%d", (z>>i)&(0x1));
    putchar('\n');
}
```

Объединяющий тип

Именованный объединяющий тип:

```
union <имя_типа>
{
<описание полей>
};
```

Тогда будут переменные типа `union <имя_типа>`.

Можно использовать `typedef`:

```
typedef union sampleUnion u12;
….
u12 myUnion; /* в main() */
```

Объединяющий тип

Указатели на объединения и доступ к полям полностью аналогичны структурам:

Если

```
typedef union sampleUnion u12;
….
/* в main() */
u12 *myUnionPtr;
myUnionPtr=(u12*)malloc(sizeof(u12));
(*myUnionPtr).a=123;
```

или

```
myUnionPtr->a=123;
```

Объединения и битовые поля

Если в объединении имеются символьное поле (**char**) и структура с битовыми полями (по одному биту), то легко получить двоичное представление кода символа (пример lect-18-03.c).

```
union charByte
{
    char ss;
    struct
    {
        unsigned char s0:1; unsigned char s1:1;
        unsigned char s2:1; unsigned char s3:1;
        unsigned char s4:1; unsigned char s5:1;
        unsigned char s6:1; unsigned char s7:1;
    } b;
};

typedef union charByte cb;
```

В **main()** объявляем **cb myChar** и выводим **myChar.b.s7 …** до **s0**.