Структуры и объединения

Алтайский государственный университет
Факультет математики и ИТ
Кафедра информатики

Барнаул 2014

Лекция 13

- Структуры
- Указатели на структуры
- Использование структур
- Объединения

Несколько заданий для самопроверки

Задание 1

- Что описывают следующие объявления?

```
void *comp() ;
void (*cmp)();
char (*(*x())[])();
```

comp – функция, возвращающая указатель на void

cmp – указатель на функцию, возвращающую void

x – функция, возвращающая указатель на массив из указателей на функции, возвращающие char

Задание 2

- Что выведет на экран следующая программа?

```
#include <stdio.h>

void main() {
   int const * p=5;
   printf("%d",++(*p));
}
```

При компиляции возникнет ошибка:
попытка изменить константное значение по указателю

Задание 3

- Что выведет на экран следующая программа?

```
#include <stdio.h>

void main() {
  int A[]={2,5,7,3,4}, *p=A+3;
  printf("%d%d",p[-2],-1[p]);
}
```

```
57
```

Задание 4

- Что выведет на экран следующая программа?

```
#include <stdio.h>
#include <string.h>
void main(){
  int count;
  char const *str="Маскарад";
  char const *ptr=str;
  ptr+=4;
  count=strlen(ptr);
  printf("%d ",count);
}
```

```
3
```

Структуры

- Что такое структура?
- Как описываются структуры?

Что такое структура?

Структура – это тип данных, представляющий собой, совокупность разнотипных переменных фиксированного размера. Каждый элемент структуры называется **полем**.

Как описывается структура?

структура

название

поля

```
struct Book {
  char author[40]; // автор
  char title[80];  // название
  int year;        // год издания
  int pages; // количество страниц
};
```

структура

название

поля

```
struct Point {
  int x; // абсцисса
  int y; // ордината
};
```

Описывается тип. Память не выделяется!

Что такое структура?

Как описываются переменные типа «структура»?

```
struct Point my_point;
struct Book book1, book2;
```

Выделяется память!

Как придать значения полям структуры?

```
my_point.x = 4;
my_point.y = 5;

strcpy ( book1.author, "А.С. Пушк
strcpy ( book1.title,  "Полтава"
book1.year = 1998;
book1.pages = 223;
```

Обращение к полям структуры – через точку

Возможна инициализация полей при описании переменной-структуры

```
struct Point O = {0,0};
struct Book book3 = {"А.С. Пушкин", "Полтава",
                     1998, 220};
```

Что такое структура?

Как описываются переменные типа «структура»?

```
struct Point {
    int x;
    int y;
} A, B={10,13};
```

Возможно совмещение описания типа и объявления переменных

```
struct {
    int x;
    int y;
} A, B={10,13};
```

Возможно описание переменных-структур безымянного типа.

Повторное описание переменных того же типа невозможно!

Что такое структура?

Как описываются переменные типа «структура»?

```
typedef struct {
    int x;
    int y;
} Point;

Point A, B={10,13};
```

Использование `typedef` позволяет укоротить описание переменных (не указывать `struct`)

```
struct Point {
    int x;
    int y;
};

struct Point A, B={10,13};
```

Без `typedef` наименование типа всегда состоит из `struct` и метки структуры

Что такое структура?

Как описываются переменные типа «структура»?

```
struct Rectangle {
    int color;
    struct Point tl; /* левый верхний угол */
    struct Point br; /* правый верхний угол */
};

struct Rectangle R1={GREEN,{10,13},{20,44}};
struct Rectangle R2;

R2.color=RED;
R2.tl.x=5;
R2.tl.y=5;
R2.br.x=50;
R2.br.x=100;
```

Структуры могут вкладываться друг в друга

Обращение к полям вложенных структур – через точку

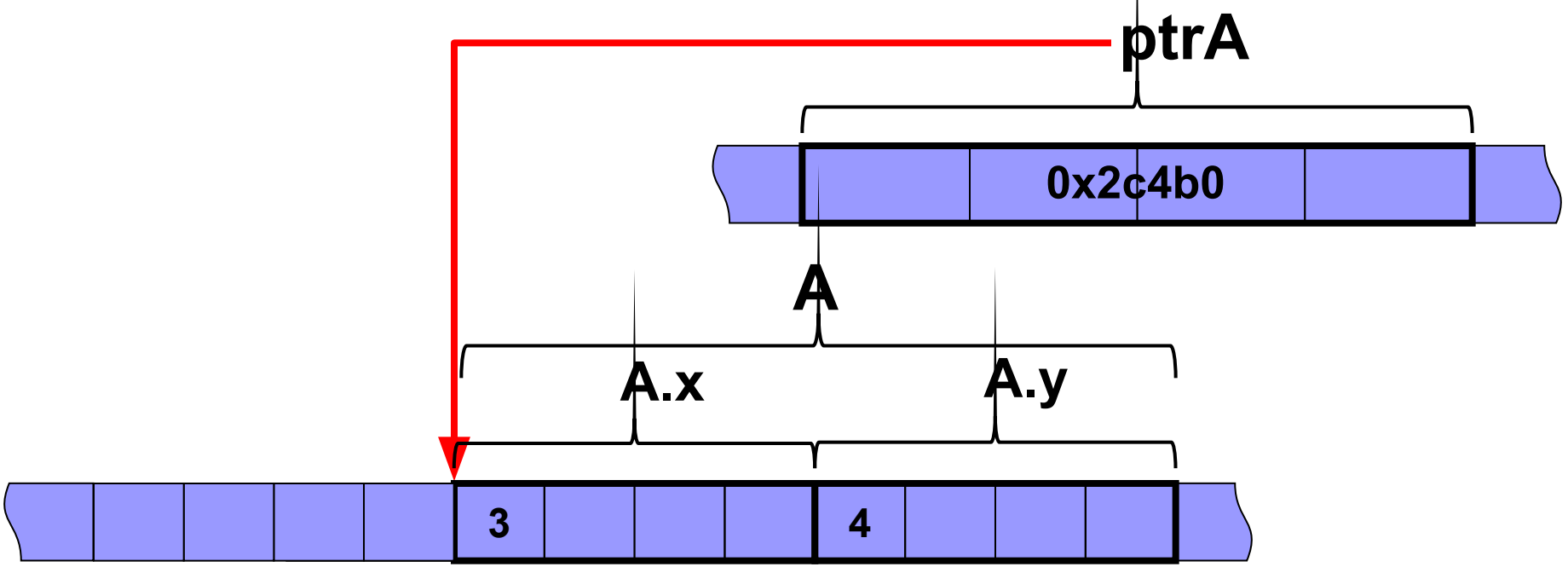
Указатели на структуры

- Указатели и структуры
- Представление структур в памяти
- Динамические структуры
- Рекурсивные структуры

Указатели на структуры

```
struct Point A = {3,4};
struct Point *ptrA = &A;
```

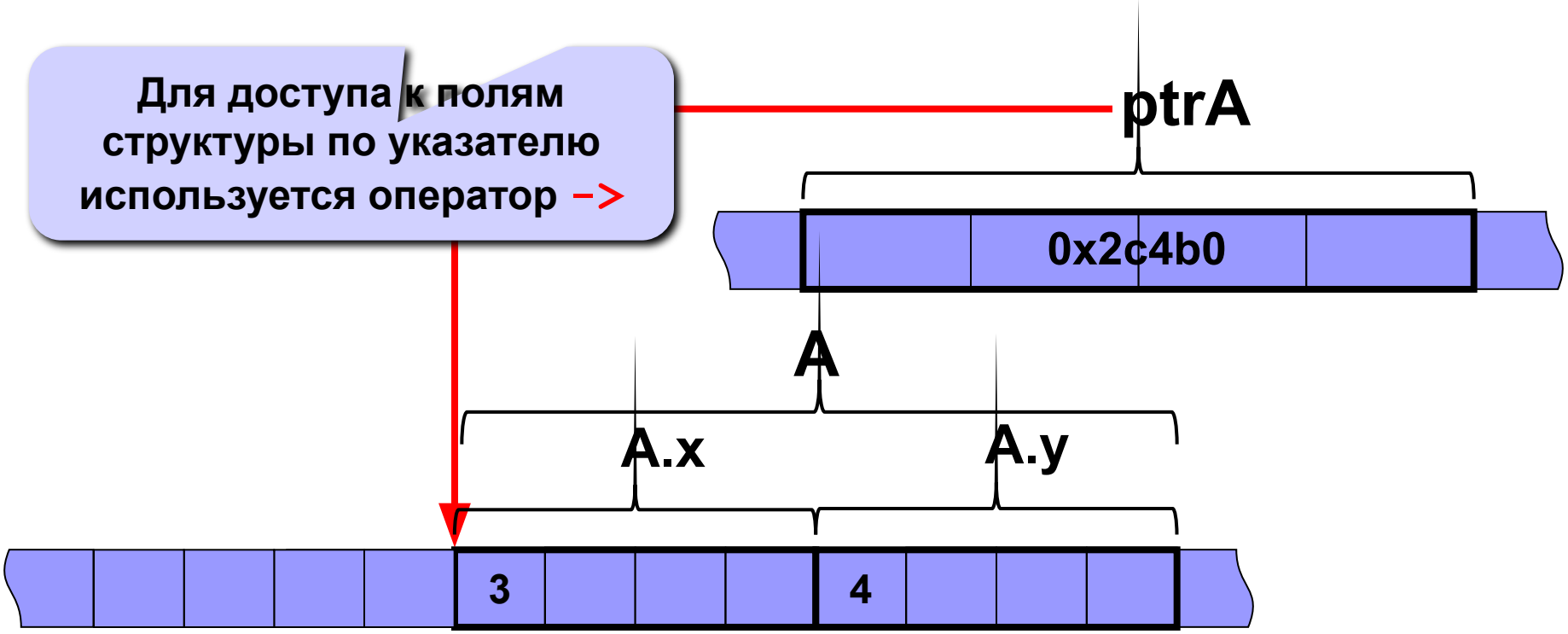
Указатель на переменную-структуру



Логическая структура != физическая структура

Указатели на структуры

```
struct Point A = {3,4};
struct Point *ptrA = &A;

ptrA->x = 5; /* то же, что и (*ptrA).x */
ptrA->y = 6; /* то же, что и (*ptrA).y */
```



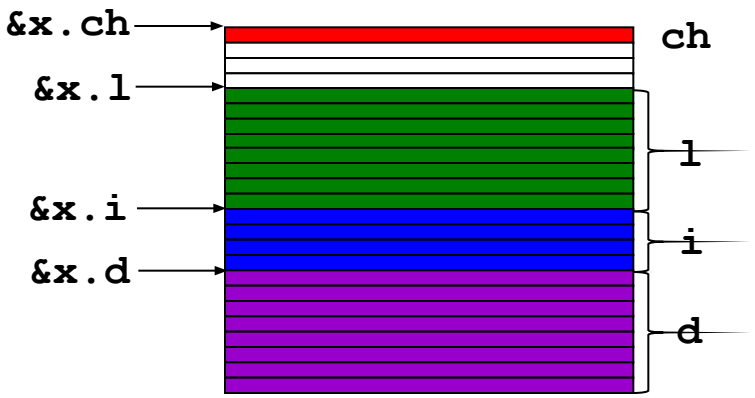
Представление структур в памяти

Логическая структура != физическая структура

```
#include <stdio.h>

struct foo {
  char ch;
  long long int l;
  int i;
  double d;
};

vo
  printf("ch:%p\n", &x.ch);
  printf("l: %p\n", &x.l);
  printf("i: %p\n", &x.i);
  printf("d: %p\n", &x.d);
  printf("%d\n", sizeof(x));
}
```

Архитектура IA32 Linux: выравнивание на границу машинного слова:
3 выравнивающих байта

```
ch:0xbfced6e0
l: 0xbfced6e4
i: 0xbfced6ec
d: 0xbfced6f0
24
```



Представление структур в памяти

Логическая структура != физическая структура

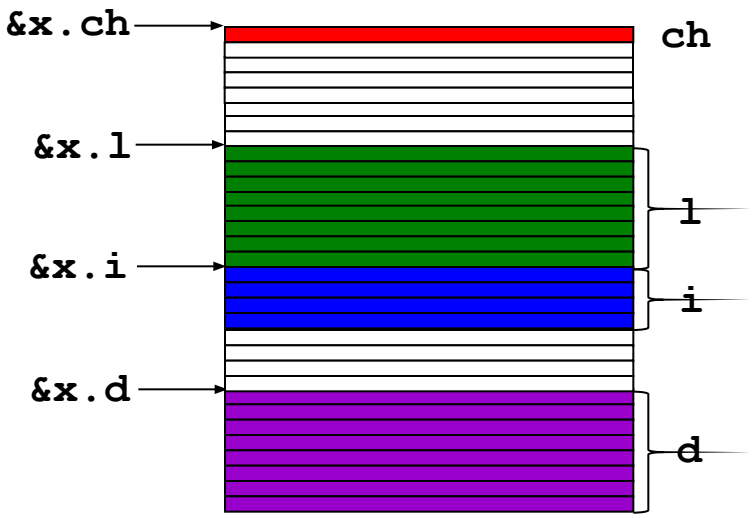
```
#include <stdio.h>

struct foo {
  char ch;
  long long int l;
  int i;
  double d;
};

vo
  struct foo x;
  p
  p
  p
  printf("%d\n", sizeof(x));
}
```

Архитектура IA32 Windows: выравнивание на границу 2 машинных слов

7 выравнивающих байт

4 выравнивающих байта

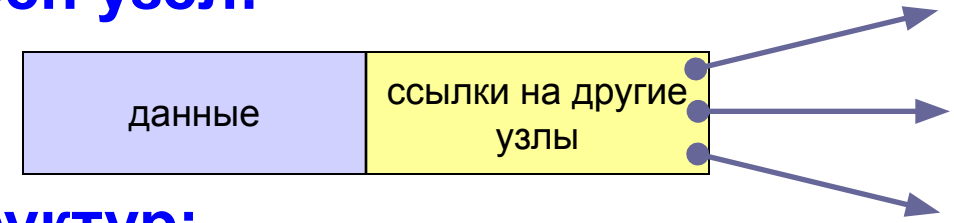
```
ch:0x22ccc0
l: 0x22ccc8
i: 0x22ccd0
d: 0x22ccd8
32
```



Динамические структуры данных

Строение: набор узлов, объединенных с помощью ссылок.

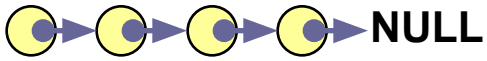
Как устроен узел:



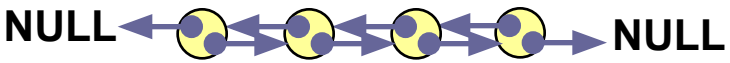
Типы структур:

списки

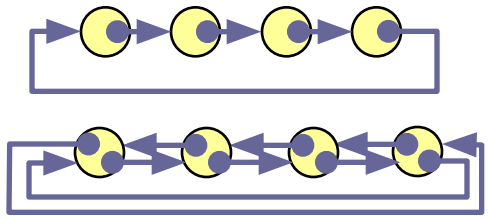
односвязный



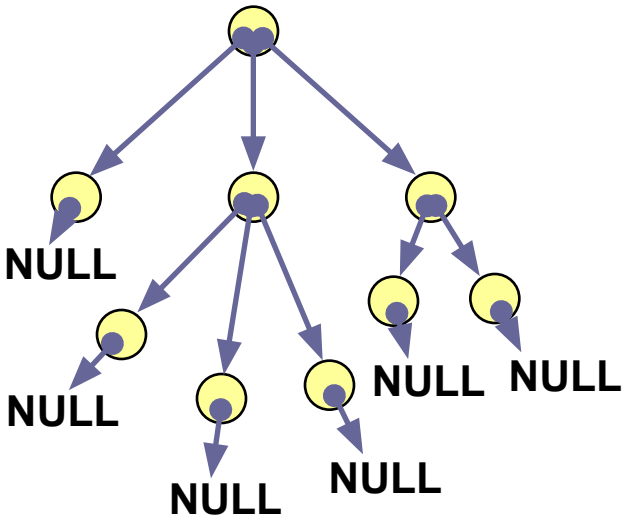
двунаправленный (двусвязный)



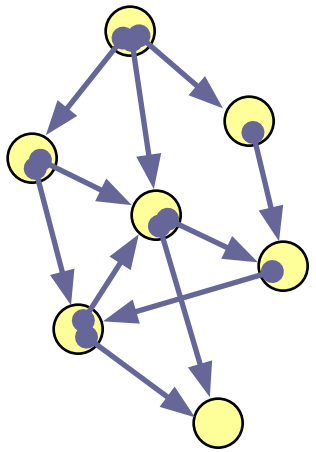
циклические списки (кольца)



деревья



графы



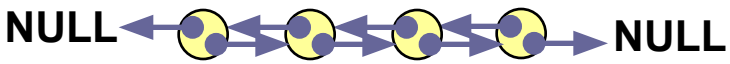
Рекурсивные структуры

При организации динамических структур данных (списков, деревьев, графов) часто используются рекурсивные структуры

Указатель на описываемую структуру

```
struct ListItem {
    int data;
    struct ListItem *prev; /* предыдущий элемент */
    struct ListItem *next; /* следующий элемент */
};
```

Двунаправленный (двусвязный) списко



Использование структур

- Копирование структур
- Массивы структур
- Динамические структуры
- Массивы структур
- Структуры и функции
- Битовые поля

Копирование структур

Задача: скопировать структуру `b1` в `b2`.

По элементам:

```
struct Book b1, b2;
...   // здесь вводим b1
strcpy(b2.author, b1.author);
strcpy(b2.title, b1.title);
b2.year = b1.year;
b2.pages = b1.pages;
```

Функция копирования строк описана в `string.h`

Копирование «бит в бит»:

```
#include <mem.h>
...
memcpy( &b2, &b1, sizeof(Book) );
```

куда | откуда | сколько байт

Первые два параметра – адреса структур!

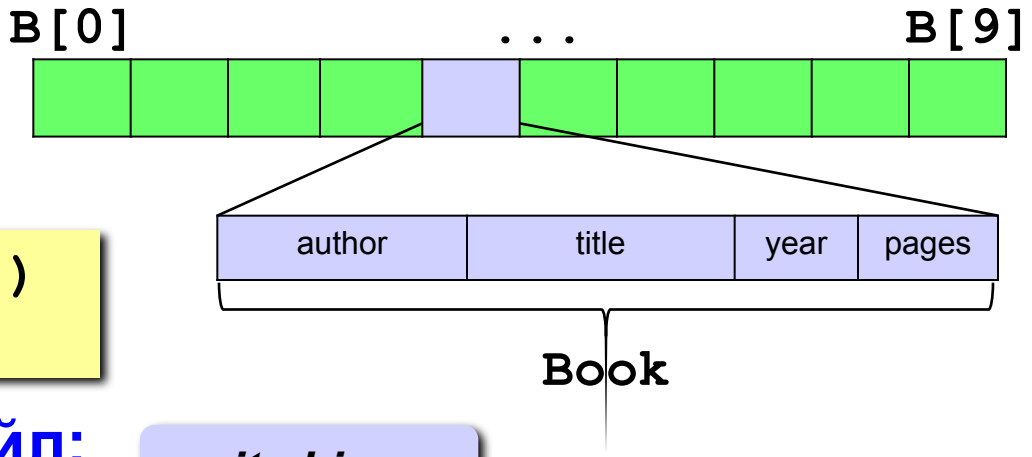
или просто так:

```
b2 = b1;
```

Массивы структур

Объявление:

```
struct Book B[10];
```



Обращение к полям:

```
for ( i = 0; i < 10; i ++ )
  B[i].year = 2008;
```

Запись в *двоичный* файл:

write binary

```
FILE *f;
f = fopen("input.dat", "wb" );
fwrite ( B, sizeof(Book), 10, f );
```

адрес массива

размер блока

сколько блоков

указатель на файл

Чтение из *двоичного* файла:

```
f = fopen("input.dat", "rb" );
n = fread ( B, sizeof(Book), 10, f );
printf ( "Прочитано %d структур", n );
```

`fread` возвращает число удачно прочитанных блоков!

Выделение памяти под стру

выделить память под структуру, записать ее адрес в переменную p

```
struct Book *p;
p = (struct Book*)malloc(sizeof(struct Book));
printf ( "Автор " );
gets ( p->author );
printf ( "Название книги " );
gets ( p->title );
printf ( "Количество страниц " );
scanf ( "%d", &p->pages );
p->year = 2008;
...
free(p);
```

освободить память

! Для обращения к полю структуры по адресу используется стрелка ->!

Динамические массивы структур

Задача: выделить память под массив структур во время выполнения программы.

```
struct Book *B;
int  n;
printf ( "Сколько у вас книг? " );
scanf ( "%d", &n );
B = (struct Book *) malloc(n*sizeof(struct Book));
...  /* здесь заполняем массив B */
for ( i = 0; i < n; i++ )
  printf ( "%s. %s. %d.\n",
          B[i].author, B[i].title,
          B[i].year);
free(B);
```

В этот указатель будет записан адрес массива

выделяем память

освобождаем память

Структуры и функции

Структуры могут быть параметрами и возвращаемыми значениями функций

```
struct Point Shift(struct Point p, int dx, int dy) {
  struct Point q;
  q.x=p.x+dx;
  q.y=p.y+dy;
  return q;
}
void main() {
  struct Point pnt={10,10};
  printf ("До сдвига: (%d,%d)\n", pnt.x, pnt.y );
  pnt=Shift(pnt,100,0);  /* сдвиг по оси X */
  printf ("После сдвига: (%d,%d)\n", pnt.x, pnt.y );
}
```

Параметры передаются по значению!

Сдвиг точки

Структуры и функции

Структуры могут быть параметрами и возвращаемыми значениями функций

```
struct Point Shift(struct Point * const p,
                   int dx, int dy)
{
  struct Point q = {p->x+dx,p->y+d
  return q;
}
void main() {
  struct Point pnt={10,10};
  printf ("До сдвига: (%d,%d)\n", pnt.x, pnt.y );
  pnt=Shift(&pnt,100,0);  /* сдвиг по оси X */
  printf ("После сдвига: (%d,%d)\n", pnt.x, pnt.y );
}
```

Большие структуры лучше передавать по указателю, чтобы избежать копирования

Битовые поля

При описании структуры можно регулировать количество памяти для полей с точностью до бита

```
struct TFriend {
  int  IsBoy;
  char IsStudent;
  char IsTheBest;
  char Age;
} friend;

friend.IsBoy     = 1;
friend.IsStudent = 0;
friend.IsTheBest = 1;
friend.Age       = 19;
friend.IsBoy     = 6;
```

Нет необходимости тратить несколько байт на бинарные значения

Иногда это не только не эффективно, но и опасно

Битовые поля

При описании структуры можно регулировать количество памяти для полей с точностью до бита

```
struct TFriend {
  int  IsBoy:1;
  char IsStudent:1;
  char IsTheBest:1;
  char Age:7;
} friend;

friend.IsBoy     = 1;
friend.IsStudent = 0;
friend.IsTheBest = 1;
friend.Age       = 19;
friend.IsBoy     = 6; /*=0*/
```

Количество бит, затрачиваемых на поле

При присвоении происходит урезание значений

Объединения

- Что такое объединение?
- Использование объединений

Что такое объединение?

Объединение – это тип данных, представляющий собой, совокупность разнотипных переменных фиксированного размера, размещаемых в одном и том же фрагменте памяти

Как описывается объединение?

объединение название

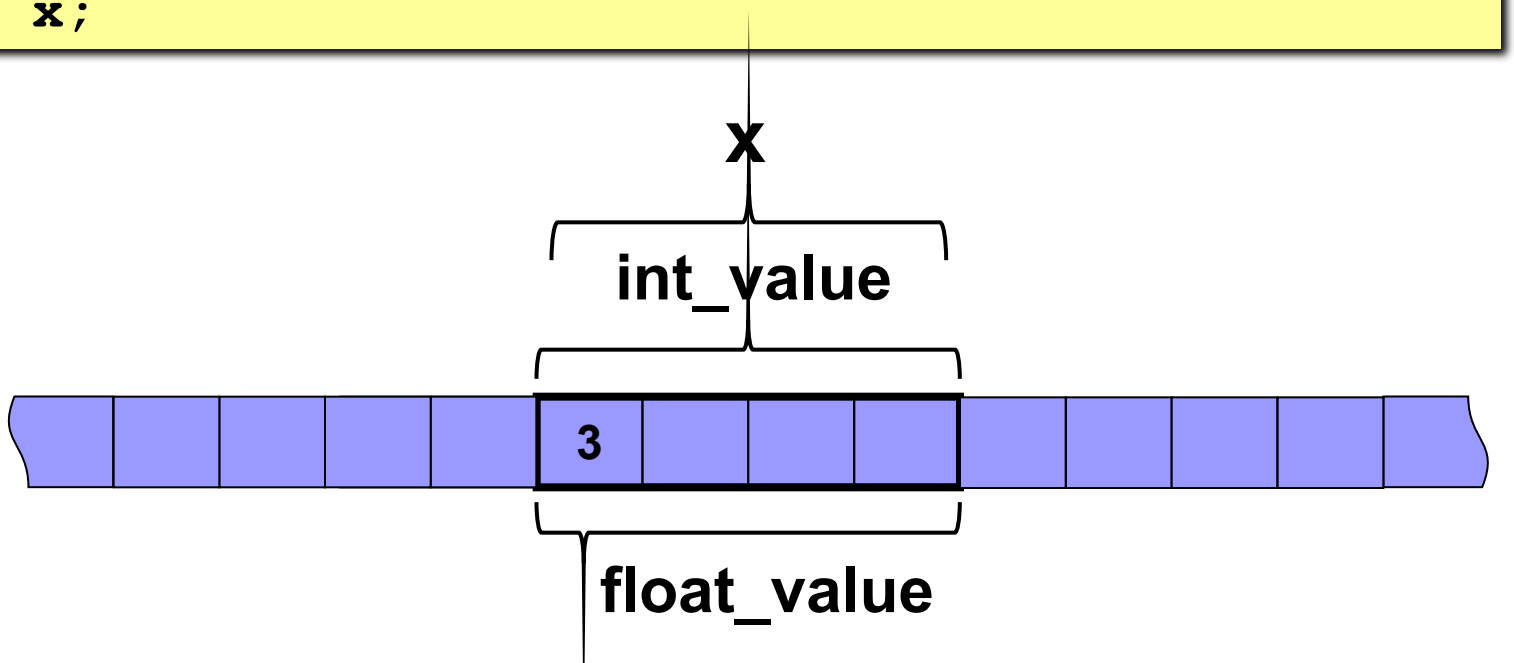
```
union int_float {
  unsigned int int_value; /* 32-битное целое */
  float float_value;      /* 32-битное ЧПТ */
} x;
```

переменная

Что такое объединение?

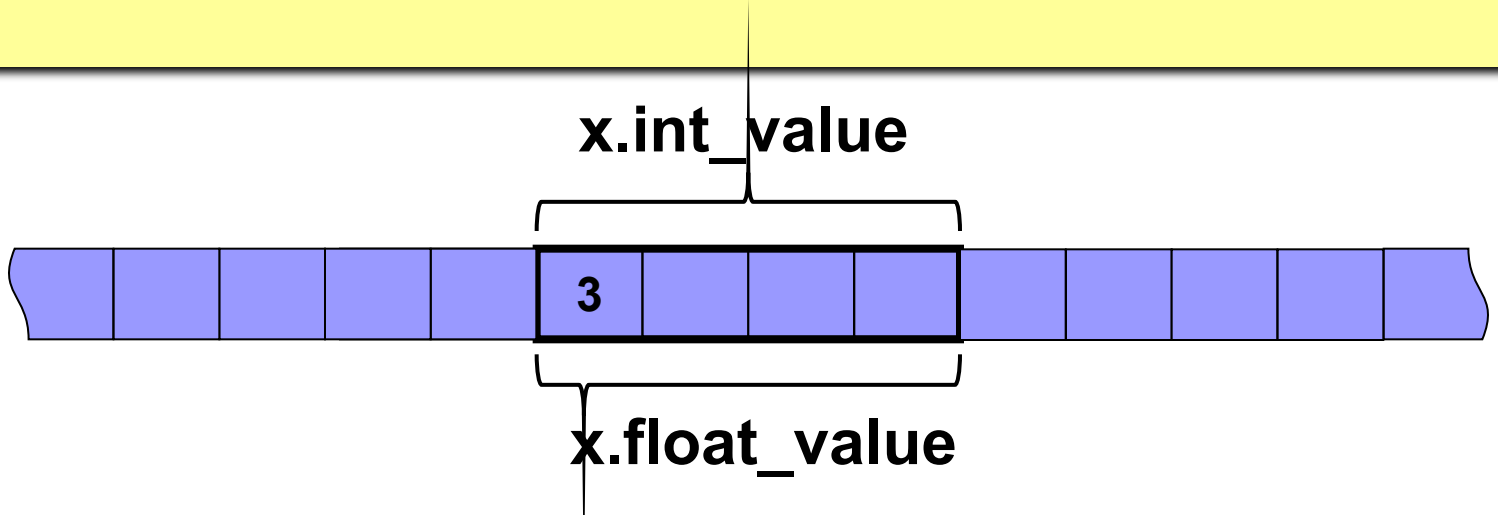
Объединение – это тип данных, представляющий собой, совокупность разнотипных переменных фиксированного размера, размещаемых в одном и том же фрагменте памяти

```
union int_float {
  unsigned int int_value; /* 32-битное целое */
  float float_value;      /* 32-битное ЧПТ */
} x;
```



Что такое объединение?

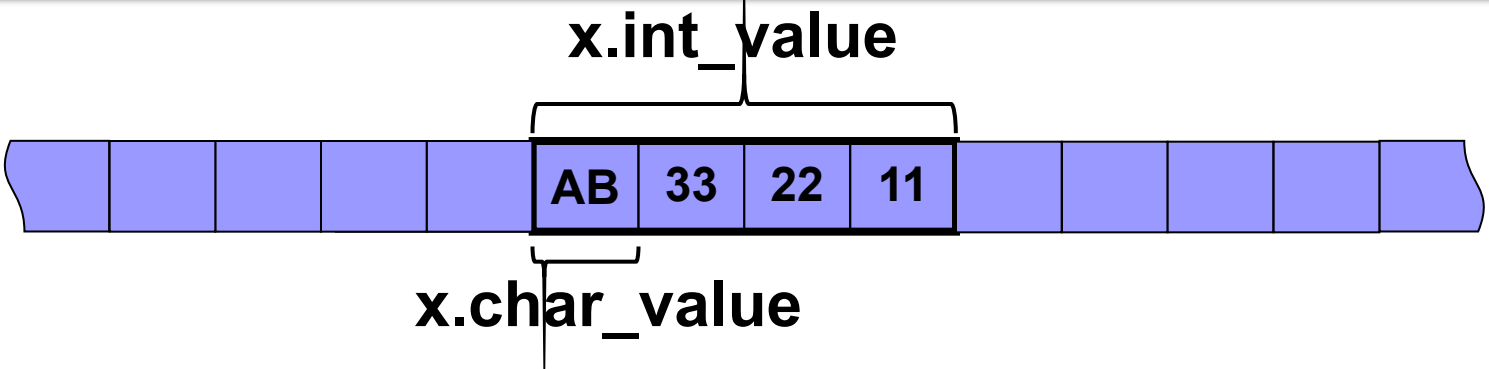
```
union int_float {
   unsigned int int_value; /* 32-битное целое */
   float float_value;      /* 32-битное ЧПТ */
} x;
...
x.float_value=3.141592654f /* float-значение*/
printf("%f\n", x.float_value); /*=> 3.14593 */

printf("%d\n", x.int_value);   /*=> 1078530011 */
printf("0x%X\n", x.int_value); /*=> 0x40490FDB */
```



Что такое объединение?

Если поля объединения различаются по размеру, то размер объединения равен максимальному из размеров полей

```
union int_char {
  unsigned char char_value;   /* 8 бит */
  unsigned int int_value;     /* 32 бит */
} x;
...
x.int_value = 0x11223344;
x.char_value = 0xAB;
printf("int:  0x%X\n",x.int_value); /*=>0x112233AB */
printf("char: 0x%X\n",x.char_value);/*=>AB */
printf("sizeof: %d\n",sizeof(x));   /*=>4 (байта) */
```



Использование объединений

Представление IP-адреса в BSD-сокетах

```
struct sockaddr_in {
  short   sin_family;      /* AF_INET for internet */
  u_short sin_port;          /* port number */
  struct  in_addr sin_addr; /* ip address */
  char    sin_zero;
};

struct in_addr {
  union {
    struct { u_char s_b1, s_b2, s_b3, s_b4 };
    struct { u_short s_w1, s_w2 };
    u_long S_addr;
  } S_un;
};
```

Использование объединений

```
struct sockaddr_in {
  short   sin_family;        /* AF_INET for internet */
  u_short sin_port;          /* port number */
  struct  in_addr sin_addr; /* ip address */
  char    sin_zero;
};

struct in_addr {
  union {
    struct { u_char s_b1, s_b2, s_b3, s_b4 };
    struct { u_short s_w1, s_w2 };
    u_long S_addr;
  } S_un;
};
...
struct sockaddr_in addr;    /* addr = "129.132.98.12" */
addr.sin_addr.S_un.s_b1 = 129;
addr.sin_addr.S_un.s_b2 = 132;
addr.sin_addr.S_un.s_b3 = 98;
addr.sin_addr.S_un.s_b4 = 12;
```

Представление IP-адреса в BSD-сокетах

Использование объединений

```
enum TSex {GIRL, BOY};
struct TFriend {
  char Name[30], Phone[30];
  TSex Sex;
  union {
    char BirthDay[20];
    char Drink[35];
  };
} friend1, friend2;
…
strcpy(frend1.Name, "Леночка");
strcpy(frend1.Phone, "902-30-...");
frend1.Sex = GIRL;
strcpy(frend1.BirthDay, "12 мая 1987");
Tfrend frend2;
strcpy(frend2.Name, "Вовик");
strcpy(frend2.Phone, "902-30-...");
frend2.Sex = BOY;
strcpy(frend2.Drink, "Только пиво");
```

Использование объединений

```
union TRect {
  struct {int Left, Top, Right, Bottom; };
  struct {int xCentr, yCentr, Width, Height;};
} rec1, rec2;

rec1.Left   = 4;
rec1.Top    = 4;
rec1.Right  = 10;
rec1.Bottom = 14;

rec2.xCentr = 7;
rec2.yCentr = 9;
rec2.Width  = 7;
rec2.Height = 11;
```

Вопросы?

- Структуры
  - Что такое структура?
  - Как описываются структуры?
- Указатели на структуры
  - Указатели и структуры
  - Представление структур в памяти
  - Динамические структуры
  - Рекурсивные структуры
- Использование структур
  - Копирование структур
  - Массивы структур
  - Динамические структуры
  - Массивы структур
  - Структуры и функции
  - Битовые поля
- Объединения
  - Что такое объединение?
  - Использование объединений

Н.Копейкин Жуков