Лекция 19

Структуры

Преподаватель Палехова Ольга Александровна,
кафедра О7 БГТУ «Военмех»

typedef-имена типов

typedef позволяет дать существующим типам данных имена-**синонимы** (*typedef*-имена, **псевдонимы**).

```
typedef имя_существующего_типа псевдоним;
```

Новый тип *typedef* не создает, любой тип может иметь множество псевдонимов

Используется для улучшения читаемости программного кода за счет

- создания «говорящих» имен типов;
- создания псевдонимов для длинных имен типов;
- упрощения сложных объявлений путем последовательного добавления имен для всё более сложных конструкций.

typedef-имена простых типов

Синтаксически объявление псевдонимов совпадает с объявлением переменных, отличие только в слове ***typedef***

Псевдонимы **для простых типов**

```
typedef unsigned int natural;
typedef unsigned int size_t;
typedef unsigned char Uint8;
typedef Uint8 byte;
typedef void * pointer;
```

natural и *size_t* – синонимы *unsigned int*

Uint8 и *byte* – синонимы *unsigned char*

pointer – то же, что и *void **

Объявления переменных:

```
natural number;
size_t size;
byte by;
natural num[100];
Uint8 *pu;
pointer p;
```

```
unsigned int number;
unsigned int size;
unsigned char by;
unsigned int num[100];
unsigned char *pu;
void * p;
```

typedef-имена типов массивов

Псевдонимы **для типов массивов**

```
typedef int Aint10 [10];
typedef double Mdouble7_9 [7][9];
typedef int *Ap_int5 [5];
typedef int (*p_Aint10) [10];
typedef char string80 [80];
```

Aint10 – то же, что *int [10]*

Mdouble7_9 – то же, что *double[7][9]*

Ap_int5 – синоним *int*[5]*

p_Aint10 – синоним *int(*)[10]*

string80 – синоним *char[80]*

Объявления переменных:

```
Aint10 a;
Aint10 matrix [8];
Mdouble7_9 m;
Ap_int5 ap = {NULL};
p_Aint10 pa = &a;
string80 text [50];
```

```
int a [10];
int matrix [8][10];
double m [7][9];
int *ap [5] = {NULL};
int (*pa) [10] = &a;
char text [50][80];
```

typedef-имена типов указателей на функции

Пример объявления функции с параметром типа указатель на функцию и типом результата указатель на функцию и указатель на такую функцию:

```
int (*f1 (int(*)(int)))(int);
int (*(*pf1)(int(*)(int)))(int);
```

прототип функции

указатель на функцию

Те же объявления с псевдонимом для типа «Указатель на функцию с параметром типа *int*, возвращающую результат типа *int*»:

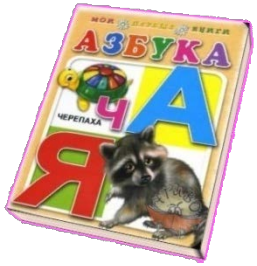
псевдоним

```
typedef int (*t_func) (int);
t_func f1 (t_func);
t_func (*pf1) (t_func);
```

прототип функции

указатель на функцию

Структуры



Свойства:
- **автор (*строка*)**
- **название (*строка*)**
- **год издания (*целое число*)**
- **количество страниц (*целое число*)**

Задача: объединить эти данные в единое целое

Свойства:
- **ФИО (*строка*)**
- **возраст (*целое число*)**
- **пол (*символ*)**
- **род занятий (*строка*)**

$$\frac{x}{y}$$

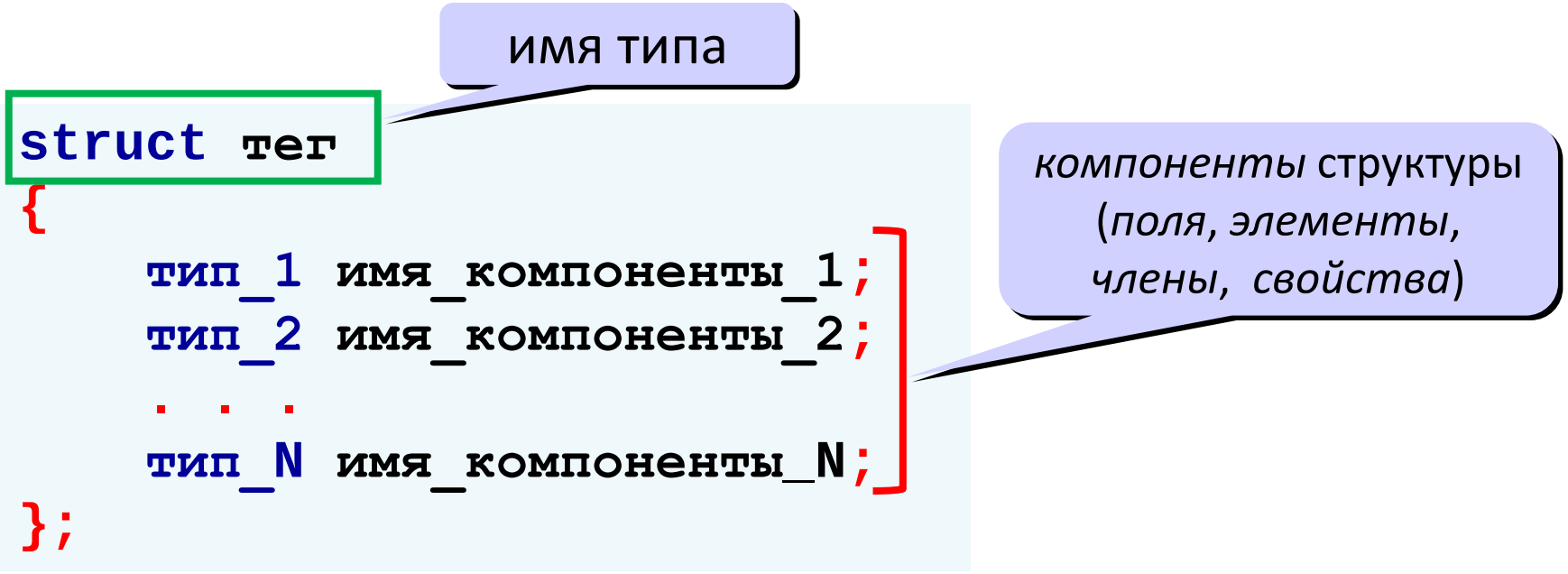
Свойства:
- **числитель (*целое число*)**
- **знаменатель (*целое число*)**

Структура представляет собой сложное данное, составленное из данных более простых типов.

Структурный тип – это производный тип данных, который объединяет в единое целое множество компонентов одного или нескольких типов.

Объявление структурного типа

Объявление структурного типа:



Особенности:

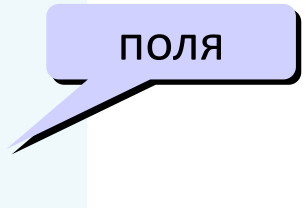
- объявляется только **тип** (память не выделяется);
- каждое поле имеет тип и имя;
- поля объявляются так же, как переменные.

Объявление структурного типа

Примеры:

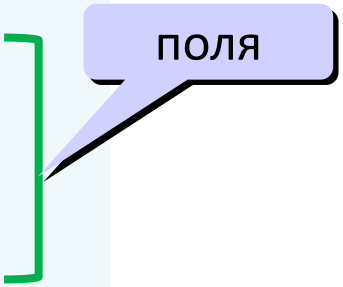
- тип Книга

```
struct book
{
    char author[50];
    char name[100];
    int year;
    int pages;
};
```



- тип Человек

```
struct person
{
    char name[60];
    unsigned char age;
    char sex;
    char occupation[30];
};
```



- тип Дробь

```
struct rational
{
    int numerator, denominator;
};
```

Объявление структурного типа

Объявление с использованием ***typedef***

- псевдоним для уже созданного типа:

```
typedef struct person human;
typedef struct person person;
```

struct person, human, person - синонимы

- задание имени для создаваемого типа:

```
typedef struct
{
    int x, y;
    int width, height;
    int color;
} rectangle ;
```

в этом случае тег не обязателен

имя типа

! **Объем памяти, выделяемой под данное структурного типа, не меньше суммы объемов памяти, требуемых для каждого поля**

Создание структур

Создание структуры

- предопределенного типа

```
struct book my_book;
struct rational r;
human man;
rectangle rect;
```

имя типа

имя типа

имя типа

имя типа

! **Переменная** структурного типа называется **структурой** или **объектом**

- одновременно с объявлением типа

```
struct complex
{
    double re, im;
} cx;
```

имя типа

имя структуры (объекта)

- без объявления типа как самостоятельной единицы

```
struct
{
    char name[60];
    char phone[11];
} subscriber;
```

идентификатор отсутствует, использовать тип повторно нельзя

имя структуры (объекта)

Поля структурного типа

Поля структур могут быть в том числе структурного типа

- предопределенного

```
struct date
{
   char day, month;
   int year;
};
struct newborn
{
    char name[60];
    struct date birthday;
    float weight;
};
```

- задаваемого в процессе объявления поля

```
struct newborn
{
   char name[60];
   struct
   {
      char day, month;
      int year;
   } birthday;
   float weight;
};
```

поле структурного типа

Операции над структурами

Операции:

- **инициализация**

```
struct book my_book = {"Гоголь Н.В.", "Нос", 2016, 64};
struct rational r = {-3, 7};
```

значения перечисляются в порядке объявления полей

- **вычисление размера**

```
sizeof (struct complex)
sizeof (rectangle)
sizeof (my_book)
```

имя типа

псевдоним типа

имя структуры (объекта)

- **получение адреса**

```
struct book * pbook = &my_book;
```

указатель на структуру

- **присваивание**

```
struct book my_book_2;
my_book_2 = my_book;
```

создается побитовая копия

- передача структуры **в функцию** в качестве **параметра**
- **возврат** значения структуры из функции через *return*

Обращение к элементу структуры

При обработке структур большинство операций выполняется над отдельными полями структуры

```
struct book my_book, * pbook = &my_book;
```

Обращение к элементам структуры:

- по имени структуры

операция доступа к элементу структуры

```
my_book.year = 2014;
```

уточненное имя

- через указатель на структуру

```
(*pbook).year = 2014;
pbook->pages = 126;
```

операция доступа к элементу структуры через указатель (операция косвенной адресации)

Обработка данных структурного типа

Уточненные имена полей структур можно рассматривать как обычные переменные соответствующих типов

Задача. Написать программу для вычисления суммы и произведения рациональных чисел.

```
#include <stdio.h>

struct rational
{
    int num,den;
};

typedef struct rational rational;

rational mult (rational, rational);
rational add (rational, rational);
void normalize (rational *);
```

объявление типа

псевдоним

нормализация дроби

Обработка данных структурного типа

```
int main()
{
    rational a, b, c;
    printf ("a=");
    scanf ("%d/%d",&a.num,&a.den);
    normalize(&a);
    printf ("b=");
    scanf ("%d/%d",&b.num,&b.den);
    normalize(&b);
    c = mult (a, b);
    printf ("a*b=%d/%d\n",c.num, c.den);
    c = add (a, b);
    printf ("c=a+b=%d/%d\n",c.num, c.den);
    c = mult (c, b);
    printf ("c*b=%d/%d\n",c.num, c.den);
    return 0;
}
```

ввод числителя и знаменателя первой дроби

нормализация первой дроби

вычисление произведения дробей

вычисление суммы дробей

Обработка данных структурного типа

Функция нормализации дроби

```
void normalize (rational *r)
{
    int a, b;
    if ( r->den < 0 )
    {
        r->den = -r->den;
        r->num = -r->num;
    }
    a = r->num > 0 ? r->num : -r->num;
    b = r->den;
    while (a && b)
        if ( a > b )
            a %= b;
        else
            b %= a;
    a += b;
    r->num /= a;
    r->den /= a;
}
```

указатель на структуру

обращение к полю через указатель

$$\frac{2}{4} \rightarrow \frac{1}{2}$$

$$\frac{-2}{-4} \rightarrow \frac{1}{2}$$

$$\frac{9}{-3} \rightarrow \frac{-3}{1}$$

Обработка данных структурного типа

Функция вычисления произведения дробей

$$\frac{3}{8}\cdot\frac{4}{9}=\frac{3}{9}\cdot\frac{4}{8}=\frac{1}{3}\cdot\frac{1}{2}=\frac{1}{6}$$

```
rational mult (rational x, rational y)
{
    rational res;
    res.num = x.num;
    res.den = y.den;
    normalize (&res);
    y.den = x.den;
    normalize (&y);
    res.num *= y.num;
    res.den *= y.den;
    return res;
}
```

структура

обращение к полю структуры

Обработка данных структурного типа

Функция вычисления суммы дробей

```
rational add (rational x, rational y)
{
    rational res;
    if (x.den == y.den)
    {
        res.num = x.num + y.num;
        res.den = x.den;
    }
    else
    {
        res.num = x.num * y.den + y.num * x.den;
        res.den = x.den * y.den;
    }
    normalize (&res);
    return res;
}
```

структура

обращение к полю структуры

Массив структур

Задача. В массив записываются результаты экзаменационной сессии студентов одной группы. Вывести список студентов, которым нужно назначить стипендию (стипендию получают студенты, сдавшие экзамены без «3»).

```
#include <stdio.h>
#define N 35

typedef struct
{
    char name[60];
    char number[10];
    int ball[4];
} student;

int main()
{
    student group[N];
    int i, j;
```

объявление типа

массив структур

Массив структур

```
int main()
{
    student group[N];
    int i, j;
    for ( i = 0 ; i < N ; i++ )
    {
        printf ("ФИО : ");
        gets (group[i].name);
        printf ("№ зачетки : ");
        gets (group[i].number);
        printf ("Оценки : ");
        for ( j = 0 ; j < 4 ; j++ )
            scanf ("%d", &group[i].ball[j]);
        getchar();
    }
    puts ("Стипендиаты:");
    for (i = 0 ; i < N ; i++ )
    {
        for ( j = 0 ; j < 4 && group[i].ball[j] >= 4 ; j++ );
        if ( j == 4 )
            printf ("%-60s%s\n", group[i].name, group[i].number);
    }
    return 0;
}
```

массив структур

обращение к полю структуры

обращение к *j*-му элементу массива оценок *i*-й структуры