Лекция 20

Структуры и объединения. Файлы

Преподаватель Палехова Ольга Александровна,
кафедра О7 БГТУ «Военмех»

Структуры

Объявление структурного типа:

```
struct тег
{
    тип_1 имя_компоненты_1;
    тип_2 имя_компоненты_2;
    . . .
    тип_N имя_компоненты_N;
};
```

имя типа

компоненты структуры (*поля*, *элементы*, *члены*, *свойства*)

Объявление переменной структурного типа:

```
struct тег имя_переменной;
```

Особенности:

- поля могут быть разных типов;
- каждое поле имеет имя;
- память выделяется для каждого поля.

Битовые поля

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

фамилия

№ зачетки

4 оценки

sizeof(student) >= 86

Проблема: для хранения одной оценки выделяется 4 байта, хотя достаточно **3 бит**.

Решение: использовать **битовые поля**.

```
typedef struct
{
    char name[60];
    char number[10];
    unsigned char ball_1:3;
    unsigned char ball_2:3;
    unsigned char ball_3:3;
    unsigned char ball_4:3;
} student;
```

поле размером 3 бита

sizeof(student) = 72

Битовые поля

Особенности:

- могут использоваться только для целочисленных данных (со знаком или без);
- количество бит может быть любым в пределах *sizeof(базовый_тип);*
- нельзя получить адрес битового поля;
- битовые поля нельзя объединять в массивы.

```
student stud;
scanf ("%d", &stud.ball_1);
```

ошибка: нельзя получить адрес битового поля

```
student stud; int ball;
scanf ("%d", &ball);
stud.ball_1 = ball;
```

используем вспомогательную переменную

Объединение

Объявление объединения:

```
union тег
{
    тип_1 имя_компоненты_1;
    тип_2 имя_компоненты_2;
    . . .
    тип_N имя_компоненты_N;
};
```

имя типа

компоненты объединения (*поля*, *элементы*, *члены*, *свойства*)

Объявление переменной типа объединение:

```
union тег имя_переменной;
```

Особенности:

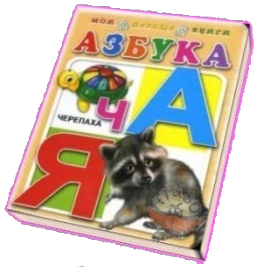
- память выделяется по размеру большего поля;
- в один момент времени может использоваться только одно поле.

! При работе с объединением нужно знать тип элемента, находящегося в нем в данный момент

Записи с вариантами

Печатные издания

Книга



Свойства:

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

Журнал

Свойства:

- **название (*строка*)**
- **номер (*целое число*)**
- **год издания (*целое число*)**
- **количество страниц (*целое число*)**

Задача: объединить эти данные в единое целое.

Идея: общие свойства (*название, год издания, количество страниц*) сделать самостоятельными полями структуры, а различающиеся (*автор, номер*) поместить в объединение.

Проблема: как узнать, данные о книге или о журнале хранятся?

Решение: использовать еще одно поле – *тип* издания.

Записи с вариантами

Задача: В книжном шкафу, рассчитанном на 100 печатных изданий, хранится некоторое количество книг и журналов. Написать программу, позволяющую выводить список экземпляров хранения отсортированным по названию и по году издания.

```
typedef struct
{
    char name[100];
    int year;
    int pages;
    char type;      /* 0 - книга, 1 - журнал */
    union
    {
        char author[50];
        int number;
    } feature;
} bookcase;
```

общие характеристики

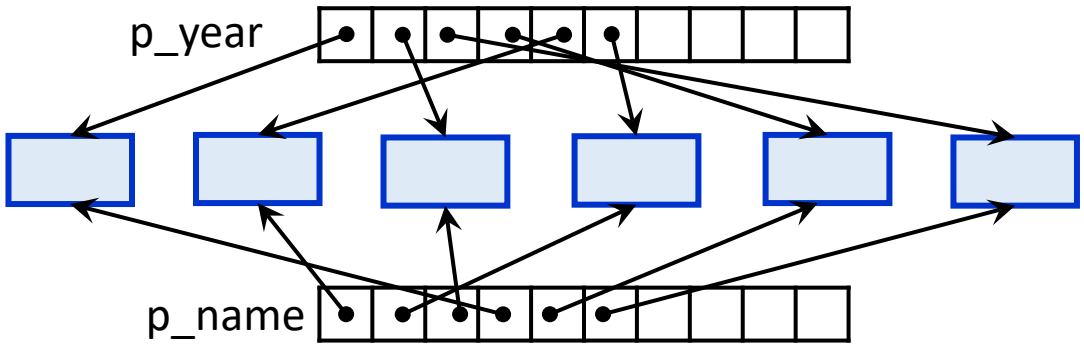
поле-объединение, будет использоваться только одно из полей, какое именно – зависит от значения поля *type*

Массив указателей на структуры

Для упорядочения элементов будем использовать **массив указателей на структуры.**

Достоинства:

- память под каждую структуру выделяется отдельно, не надо искать большой блок, в который поместятся все записи;
- при сортировке элементов не надо копировать большие объемы данных, достаточно перенастроить указатели;
- для упорядочения по разным полям можно использовать два массива указателей, не надо будет выполнять сортировку при выборе другого порядка вывода.



Задача о печатных изданиях

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 100

typedef struct
{
    char name[100]; /* название */
    int year;       /* год издания */
    int pages;      /* число страниц */
    char type; /* 0 - книга, 1 - журнал */
    union           /* у книги автор, у журнала номер */
    {
        char author[50];
        int number;
    } feature;
} bookcase;

/* функция создания и заполнения структуры */
bookcase * input_thing (void);
/* функция добавления структуры в упорядоченный массив
первый параметр - добавляемый элемент,
второй и третий - массив и его размер,
последний - указатель на функцию сравнения */
void add_thing (bookcase *, bookcase *[], int,
                int(*)(bookcase*,bookcase*));
/* функция сравнения названий */
int cmp_name (bookcase*,bookcase*);
/*функция сравнения годов издания */
int cmp_year (bookcase*,bookcase*);
/* функция печати списка */
void print (bookcase *[], int);
```

объявление структурного типа является глобальным,

функция возвращает указатель на структуру

параметр функции – массив указателей на структуры

указатель на функцию, параметры которой – указатели на структуры

Задача о печатных изданиях

```
int main()
{
    bookcase *p_name[N] = {}; /* массив для упорядочивания по названию */
    bookcase *p_year[N] = {}; /* массив для упорядочивания по году */
    int n = 0;  /* размер массива */
    int menu;
    do
    {
        system ("cls");
        puts ("1 - Добавить экземпляр");
        puts ("2 - Вывести список");
        puts ("3 - Выход");
        scanf ("%d", &menu);
        switch (menu)
        {
            case 1 :
            {
                bookcase *tmp = input_thing(); /* создаем и заполняем структуру */
                /* размещаем указатель на нее в массиве, упорядоченном по названию
                add_thing (tmp, p_name, n, cmp_name);
                /* размещаем указатель на нее в массиве, упорядоченном по году */
                add_thing (tmp, p_year, n, cmp_year);
                n++;    /* увеличение счетчика структур */
                break;
            }
            case 2 :
                puts ("1 - по названиям");
                puts ("2 - по году издания");
                {
```

массивы указателей на структуры

выбор действий осуществляется с помощью меню

последний пункт – «Выход», пока он не выбран, программа продолжает работать

Задача о печатных изданиях

```
            }
            case 2 :
                puts ("1 - по названиям");
                puts ("2 - по году издания");
                {
                    int menu;
                    scanf ("%d%*c", &menu);
                    switch (menu)
                    {
                        case 1 :    /*  печать в алфавитном порядке названий
                            print (p_name, n); break;
                        case 2 :    /* печать по возрастанию лет издания */
                            print (p_year, n); break;
                        default : puts ("Некорректно");
                    }
                    getchar();
                }
        }
    }
    while (menu != 3);
    while ( n>0 )
        free (p_name[--n]); /* освобождение памяти от всех структур */
    return 0;
}
```

вложенное меню

выход из цикла меню при выборе пункта «Выход»

перед выходом из программы освобождаем память от всех данных

Задача о печатных изданиях

Функция вывода списка на экран. Входные данные – массив указателей на структуры и его размер.

массив указателей на структуры

шапка таблицы, по ней осуществляем расчет ширины столбцов

```
void print (bookcase * p[], int n)
{
    int i;
    puts ("   Тип   | Автор/Номер |    Название      | Год  | Страниц");
    puts ("-----------------------------------------------------------");
    for ( i = 0 ; i < n ; i++ )
    {
        printf (" %-7s| ",p[i]->type?"журнал":"книга");
        if (p[i]->type)
            printf("%11d",p[i]->feature.number);
        else
            printf("%-11.11s",p[i]->feature.author);
        printf (" | %-15.15s | %4d | %6d\n",
                p[i]->name, p[i]->year, p[i]->pages);
    }
}
```

выбираем поля объединения в зависимости от типа издания

При выводе строк выравнивание по левому краю

Задача о печатных изданиях

Функция создания и заполнения структуры. Выходное данное – адрес созданной структуры.

```
bookcase * input_thing (void)
{   /* выделение памяти под структуру */
    bookcase * t = malloc (sizeof(bookcase));
    char *tmp;
    int c;
    printf ("Тип издания (0 - книга, 1 - журнал) : ");
    do
    {
        scanf ("%d%*c", &c);
    }
    while ( c != 0 && c != 1 );
    t->type = c;          /* сначала определяем тип издания */
    if ( c ) /* в зависимости от типа заполняем одно из полей объедин
    {
        printf ("Number : ");
        scanf ("%d%*c", &t->feature.number);
    }
    else
    {
```

выделение памяти для структуры

тип издания только 0 или 1

t – указатель, поэтому ->, feature – НЕ указатель, поэтому .

Задача о печатных изданиях

```
if ( c ) /* в зависимости от типа заполняем одно из полей объединения *
{
    printf ("Number : ");
    scanf ("%d%*c", &t->feature.number);
}
else
{
    printf ("Author : ");
    fgets (t->feature.author, 50, stdin);
    tmp = strchr (t->feature.author, '\n');
    if ( tmp ) *tmp = '\0'; /* убираем переход на новую строку */
    else while (getchar()!='\n');  /* если строка была слишком длинной,
                                      /* то очищаем поток ввода */
}
printf ("Name : ");
fgets (t->name, 100, stdin);
tmp = strchr (t->name, '\n');
if ( tmp ) *tmp = '\0';
else while (getchar()!='\n');
printf ("Год издания : ");
scanf ("%d", &t->year);
printf ("Число страниц : ");
scanf ("%d%*c", &t->pages);
return t;   /* возвращаем адрес структуры */
}
```

читаем не больше символов, чем помещается в поле

если строка короткая, в ней будет символ ’\n’, его надо убрать, если строка слишком длинная, то ее «хвост» надо удалить из потока ввода

после ввода числа убираем ‘\n’ из потока ввода

Задача о печатных изданиях

Функция вставки элемента в массив.

Входные данные:

t – адрес добавляемой структуры;

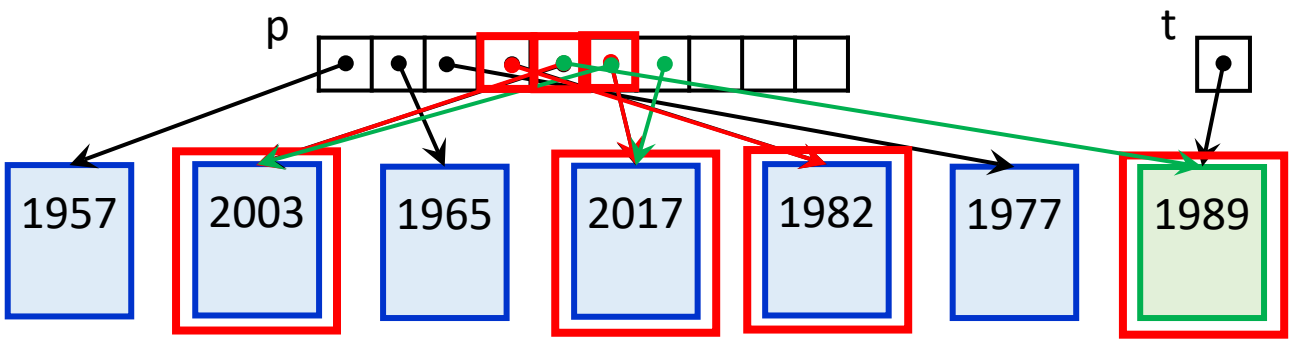
p – массив указателей;

n – текущий размер массива;

cmp – указатель на функцию сравнения структур.

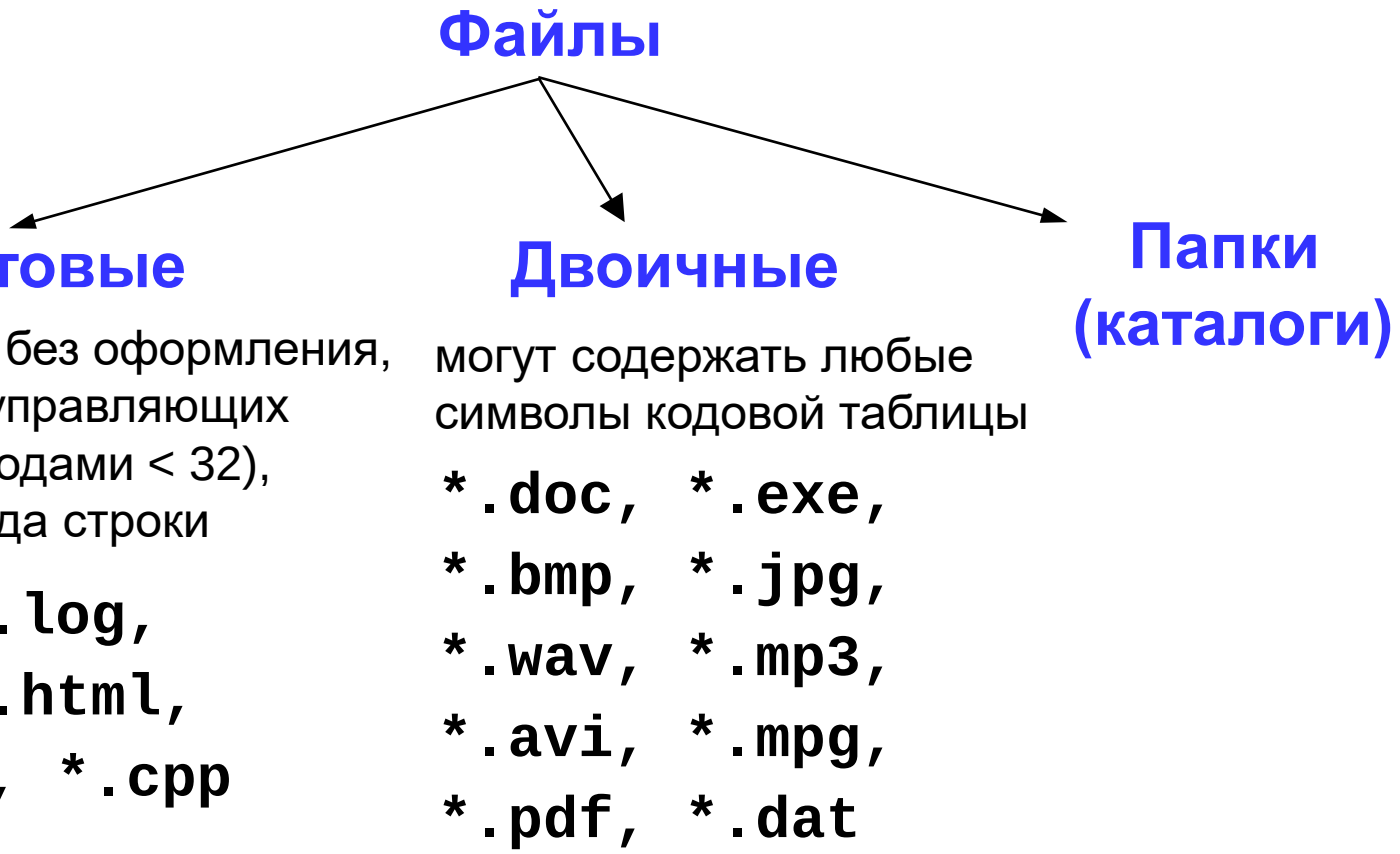
```
void add_thing (bookcase *t, bookcase *p[], int n, /* что добавляе
                int (*cmp)(bookcase*,bookcase*))    /* как сравни
{
    for ( n-- ; n >= 0 && cmp (p[n], t) > 0 ; n-- ) /* поиск места
            p[n+1] = p[n];                          /* сдвиг элементов */
    p[n+1] = t;      /* непосредственн
}
```

поиск места вставки со сдвигом элементов



Файлы

Файл – это область на диске, имеющая имя.



Потоки

Язык Си содержит средства **низкоуровневой** и **высокоуровневой** обработки файлов. Высокоуровневая обработка осуществляется с помощью **потоков**.

Поток – это источник или получатель данных, связанный с устройством внешней памяти или с каким-то другим внешним устройством.

Потоки:

- **текстовый** – это последовательность символов, объединенных в строки, разделенные символом ***'\n'*** (в одних ОС один символ (*LF*, код 10), в других два символа (*CR LF*, коды 13 10)).
- **бинарный** – это последовательность непреобразованных байтов.

Для использования потоков требуется включение заголовочного файла **stdio.h**

Создание потока

Объявление потока:

```
FILE * имя_переменной;
```

структура, содержащая всю информацию, необходимую для управления потоком

Альтернативные названия: *поток, указатель на поток, файловая переменная, указатель на файл, файловый указатель, логический файл*

Когда программа начинает работу, уже открыты три потока: **stdin** (связан с клавиатурой), **stdout** и **stderr** (связаны с монитором).

Для открытия других потоков используется функция

```
FILE * fopen ( char * , char * );
```

имя файла

режим открытия

Режимы открытия потоков

Основные режимы открытия:

Основные режимы открытия создают **текстовые** потоки

- **"w"** (*write*) – создать файл и открыть для записи. Если файл не существует, он будет создан. Если файл существует, то он будет удален и создан новый файл;
- **"r"** (*read*) – открыть файл для чтения. Файл должен существовать;
- **"a"** (*append*) – открыть файл для дополнения. Если файл не существует, он будет создан. Если файл существует, запись будет осуществляться в конец файла.

При невозможности открыть файл в указанном режиме функция *fopen()* возвращает **NULL**

Режимы открытия потоков

Расширенные режимы открытия:

- **"w+"** – создать файл и открыть его для записи и чтения;
- **"r+"** – открыть файл для чтения и редактирования;
- **"a+"** – открыть файл для дополнения и чтения.

Для **бинарных** потоков режимы открытия дополняются буквой ***b:***

- **"wb"**, **"rb"**, **"ab"**
- **"w+b"**, **"r+b"**, **"a+b"** или **"wb+"**, **"rb+"**, **"ab+"**

Существующий поток можно перенаправить на файл:

```
FILE * freopen ( char *, char *, FILE *);
```

После завершения работы с потоком его надо закрыть:

```
int fclose ( FILE * );
```