

Технология ООП

Санкт-Петербургский государственный политехнический университет

13 сентября 2011

Еще немного об указателях

```
double *p1, *p2;  
*p1=12.56e7;  
*p2=8.85e12;
```

```
double *p1,*p2;  
p1 = new double;  
p2 = new double;  
*p1=12.56e7;  
*p2=8.85e12;
```

Константные указатели и указатели на константу

```
const char *title = "Diagram"; // Указатель на  
    константу  
char *const title="Diagram"; // Константный  
    указатель
```

Константные указатели и указатели на константу

```
const char *title = "Diagram"; // Указатель на  
    константу  
char *const title="Diagram"; // Константный  
    указатель
```

Константные указатели и указатели на константу

Что будет если выполнить:

```
title[0] = 'd';  
title = new_str;
```

В первом и во втором случае?

```
char* strcpy (char* dest , const char* source);
```

Способы передачи параметров

В языках C и C++, в отличие от многих других языков, передача параметров осуществляется по значению — это значит что в функции происходит работа с копией переменной.

```
void func(int x);  
func(value);
```

```
void func(float *x);  
func(array);
```

```
void func(float x[]);  
func(array);
```

```
void func(int &out_x);  
func(value);
```

```
double array[100];  
  
double& getV(int i) { return array[i]; }  
  
getV(1) = 5.0;  
printf("%f\n", getV(1));
```

Указатель на любую функцию, которая возвращает вещественное число и требует одно вещественное число в качестве параметра:

```
double (*pFunc)(double);
```

```
#include <math.h>

double (*pFunc)(double);

if(input)
    pFunc = sin;
else
    pFunc = cos;

double val = pFunc(1.0);
```

Языки C и C++ позволяют определить новый тип переменных, присвоив символьное имя какому-то заданному типу.

```
typedef int  points;  
typedef struct Node*  NodePtr;  
typedef char Msg[100];
```

```
struct Man
{
char *Name; // Имя
int Age; // Возраст
};
```

```
struct Man m;  
m.Name = new char[100];
```

//TODO: Написать сортировку

```
Man ar[3] = {  
    {"Man1", 60},  
    {"Man2", 50},  
    {"Man3", 80}  
};
```

Union — это структура, все элементы которой имеют нулевое смещение.

```
struct Test
{
int selector;
union
{
long long_value;
short short_value;
}
};
```

Структура позволяет работать с битовыми полями.

<целый тип> [идентификатор] : <размер поля>;

```
struct byte
{
    char a: 1;
    char b: 1;
    char c: 1;
    char d: 1;
    char e: 1;
    char f: 1;
    char g: 1;
    char h: 1;

};
```

```
union chbyte
{
    byte bit;
    char x;
} b;
```

```
b.x = 'a';
printf("%1c\n", b.bit.h ? '1' : '0');
```

```
typedef enum
```

```
{  
One, Two, Three  
} Digits;
```

```
enum
```

```
{  
One = 1, Two, Three  
} digit;
```

```
digit = Two;  
Digits digit2 = digit;
```

Рекурсивная функция — функция вызывающая саму себя.
Функция вычисляющая число Фибоначчи:

```
unsigned long fib (int n)
{
unsigned long f=0;
if (n>0)
if (n==1)
  f = 1;
else
  f = fib(n-1) + fib(n-2);
return f;
}
```

Операции с файлами

```
char *fn = "test.txt"; // Имя файла  
FILE *fp; // Указатель на файловую структуру  
if(fopen (fn , "rt") == NULL) printf("File_not_found  
    \n");
```

- r — Открыть существующий файл только для чтения.
- r+ — Открыть существующий файл для обновления.
- w — Создать файл для записи.
- w+ — Создать новый файл для обновления.
- a — Открыть для записи в конец файла.
- a+ — Открыть для чтения и записи в конец файла.
- b — Открыть файл в двоичном режиме.
- t — Открыть файл в текстовом режиме.

```
char str[100];  
  
while(!feof(fp))  
{  
    fgets(str, 100, fp);  
}  
fclose(str);
```

Параметры командной строки

- `argc` — количество параметров в командной строке программы.
- `argv` — массив из строк содержащий переданные параметры.

```
void main (int argc , char *argv [])  
{  
    char src_filename[20], dst_filename[20];  
    if(argc < 3)  
        printf("error\n");  
    strncpy(src_filename , argv[1] , 20);  
    strncpy(dst_filename , argv[2] , 20);  
    //TODO: написать копирование файлов  
}
```

Базовые понятия которые имеют существенную роль в ООП:

- инкапсуляция
- наследование
- полиморфизм