程序结构*Program Structure*

存储类关键字*storage class specifier*

基本概念

- 包括：**auto，extern，static，register，typedef等**
- 声明变量时，最多使用一个存储类关键字

  > `typedef register int x;`编译不能通过

- 分类：
  - 从变量的作用域角度（空间）来分，可以分为**全局变量** 和**局部变量**
  - 从变量的存在时间角度（生存期）来分，可以分为**静态存储方式**和**动态存储方式**
- 修饰关系：直接修饰变量，而不是数据类型。

  > static int p　*a static pointer to integer，换言之static先修饰p*

典型例题

C语言中静态变量和外部变量的初始化是在___阶段完成的。

> 编译

`sizeof(int)`可计算整型所占的内存字节数，但是`sizeof( )`并不是一个函数，而是一个运算符（操作符，operator）。

> √

C语言中定义的全局变量存放在堆区，局部变量存放在栈区。

> x

函数指针

- 注意区分 ***function prototype*** 和 ***definition***

定义函数指针*function pointer*

```
int *(*p)(char *,double);
```

> 理解：()优先级最高，(*p)表明是指针

写出函数指针，函数返回类型为***void***，参数有两个：一个是***interger variable***,一个是 ***a pointer to an array of 10 integers***

```
void (*p)(int,int(*q)[10]);
```

> 这道题坑主要在第二个参数

返回类型为函数指针的函数原型*function prototype*

写出func函数原型，参数表是***int***，返回一个pointer to the function ***void f(int n)***

```
void(*func(int))(int)
```

> 理解：最内层()优先级最高，func()表明是函数。*func表明返回类型是指针
>
> 注意区别`void (*func) (int)`,这是一个指针

函数参数为函数指针

```
int f(int a) {return a;}
printf("%d",g(1,f));

int g(int c,int f(int)) {
  return f(c);
}

int g(int c,int (*f)(int)) {
  return (*f)(c);
}
//以上两种写法都是对的,即函数名称作为参量时
//既可以直接引用，也可以当作指针
```

`[] () *`等运算符优先级

以下哪个选项中的`p`是指针：

```
int* *p();
```

```
int *p();
```

```
int (*p)[5];  √
```

```
int *p[6];
```

typedef

- 一般数据类型起别名

```
typedef int myint;
typedef struct node node;
```

- 函数指针的typedef

```
typedef int *(*T)(char *,double);//语法完全类似于定义函数指针
T p;//定义函数指针
```

#define

- 只是复制粘贴

```
#define char* type1
typedef char* type2
type1 s1,s2;
type2 s3,s4;
```

等价于

```
char* s1,s2;//s2不是指针
char *s3,*s4;
```

- 还有乘法运算没括号，此处略

标准头文件结构

```
#ifndef _LINKLIST_
#define _LINKLIST_
#endif
```

递归、搜索与排序、复杂度