千锋嵌入式学院C语言培训

指针2

内容摘要

- 指针的数组
- 指向数组的指针
- 指向指针的指针
- 指向函数的指针
- 动态内存分配
- 指针与结构体
- 函数指针与结构体

指针的数组

- 数组的元素类型是指针类型

int * ptrs[10];

数组长度是10

数组名称是 ptrs

int * 表示数组元素的类型是指向整数的指针

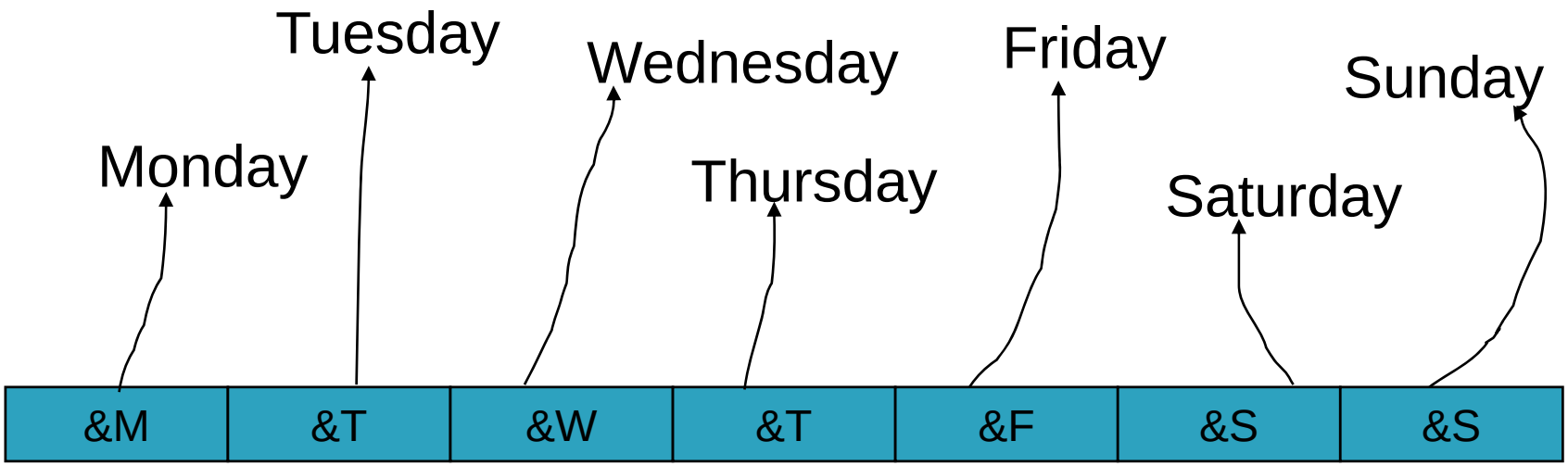
ptrs是一个数组，元素类型是指针，共10个指针

指针的数组

```
char *array[] = {
        "Monday", "Tuesday", "Wednesday","Thursday",
        "Friday", "Saturday", "Sunday"
};

int main()
{
        int i = 7;
        for(i=0;i<7;i++)
                printf("%s\n", array[i]);
        return 0;
}
```

指针的数组


Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
&M
&T
&W
&T
&F
&S
&S

指针数组

```
 1 #include<stdio.h>
 2
 3 int main(int argc, char *argv[])
 4 {
 5     int i=0;
 6
 7     for(i=0; i<argc; i++)
 8     {
 9         printf("argv[%d]:%s\n", i, argv[i]);
10     }
11
12     return 0;
13
14 }
```

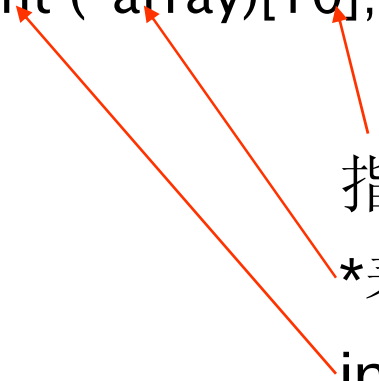
指针数组

- 编写程序，接收用户从命令行输入的多个整数，求和后打印出结果

cmd 0 1 2 3
6
cmd 0 1 2 3 4 5
15

指向数组的指针

- int (*array)[10];



指向的数组长度是10

*表示array是指针类型的变量

int 表示指向的数组元素类型

array 是一个指针，指向长度为10的整形数组

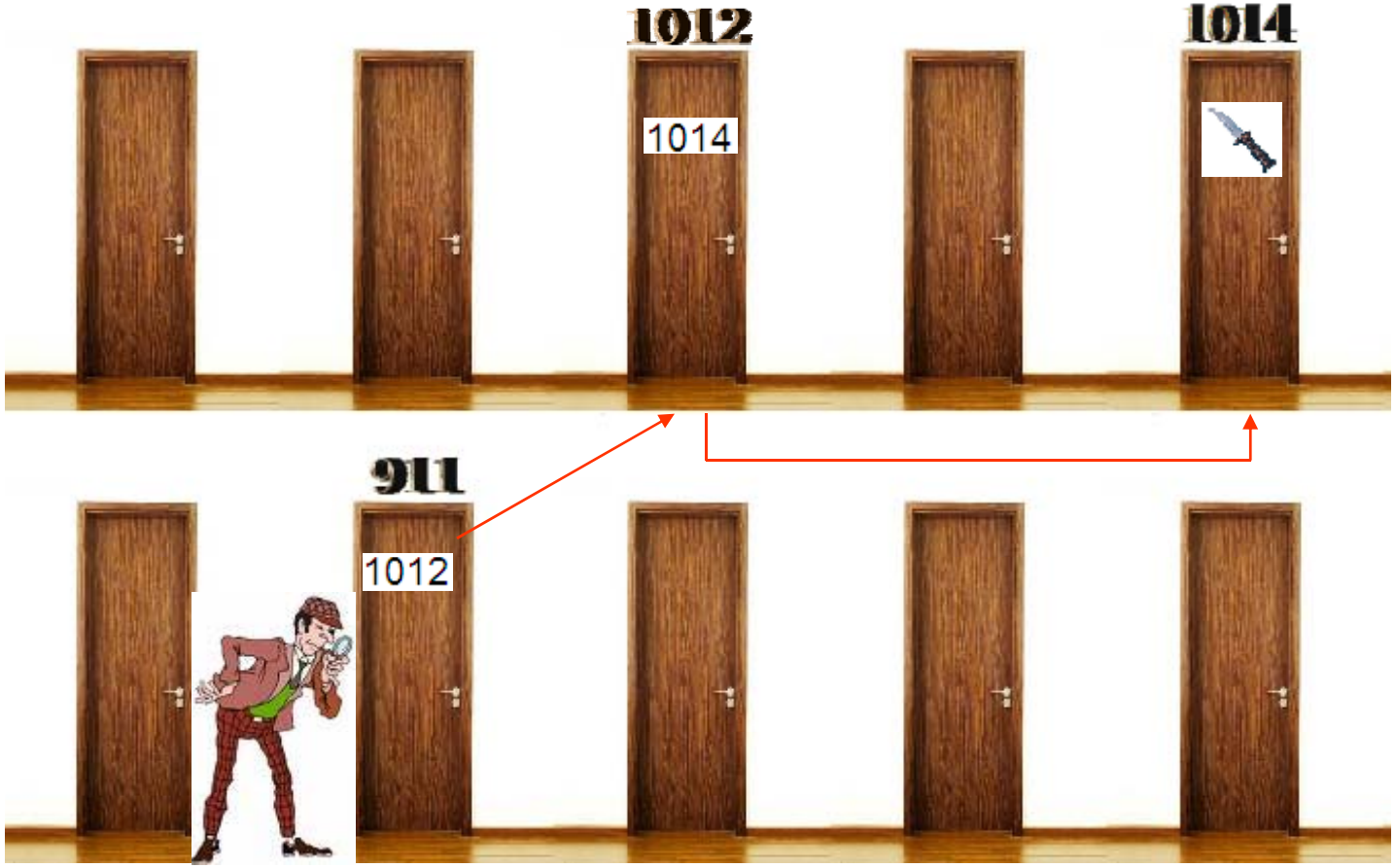
(*array)必须加括号，括号[]运算符优先级高于*

指向数组的指针

```
int sum( int (*array)[10])
{
        int i=0;
        int s = 0;
        for(i=0;i<10;i++)
                s += (*array)[i];
        return s;
}

int main()
{
        int nums[10];
        int i = 0;
        for(i=0;i<10;i++)
                nums[i] = i;
        printf("%d\n", sum(&nums));
        return 0;
}
```

指向指针的指针

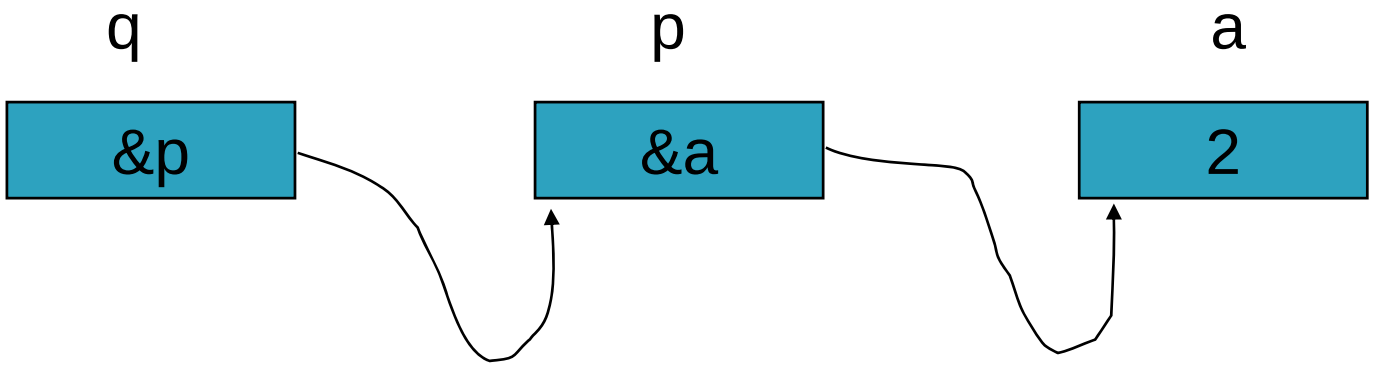


源自清华 值得信赖

指向指针的指针

- 一种特殊的指针，指向的变量类型是指针

```
int a = 2;
int *p = &a;/*p是指向a的指针*/
int **q = &p;          /*q是指向p的指针*/
```



指向指针的指针

```
int main()
{
        int a = 2;
        int *p = &a;
        int **q = &p;
        printf("&a = %x, p = %p, *p = %d\n", &a, p, *p);
        printf("&p = %x, q = %p, *q = %x\n", &p, q, *q);
        printf("a = %d, *p = %d, **q = %d\n", a, *p, **q);

        return 0;
}
```

- 常见错误:
- int main(void)
- {
- int ** p;
- int a;
- *p=&a;
- return 0;
- }

指向指针的指针

- void func(char array[14]) 等价于 void func(char *parray)
  - parray指向数组的第一个元素
- void func(char *array[14]) 等价于 void func(char **parray)
  - parray是一个指向指针的指针，指向数组的第一个元素

```
int main(int argc, char *argv[]) /* equals to int main(int argc, char **argv) */
{
        int i = 0;
        for(i = 0;i < argc;i++)
                printf("%s ", *(argv + i)); /* *(argv + i) is a pointer */
        printf("\n");
        return 0;
}
```

指向指针的指针

- 练习：自定义一个函数

```
int main()
{
        int a = 1, b = 2;
        int *p = &a;
        p = &b;             /* 定义等价函数 */
        /* 调用定义出来的函数 */
        /* 调用完成后 p 指向 b */
        /* 即 *p = 2, a = 1,  b = 2 */
}
```

指向指针的指针

- 练习：自定义一个函数

```
int main()
{
        int a = 1, b = 2;
        int *p = &a, *q = &b;
        int *t = p;         /* 写一个等价的函数 */
        p = q;
        p = t;
        /* 执行完成后交换p和q */
        /* 即 *p = 2, *q = 1, a = 1,  b = 2 */
}
```

指向函数的指针

- int (*pfunc) (int a, int b); /*声明一个函数指针变量pfunc*/

指向的函数有两个int类型参数

指针变量名字是pfunc

*表示这是一个指针

指向的函数返回值是int类型

- int * func(int a, int b); /* 声明一个函数 */
  /* 函数返回值类型是一个指向int变量的指针 */

指向函数的指针

```
void hello()
{
        printf("Hello World!\n");
        return ;
}

int main()
{
        void (*pfunc)() = hello;  /* pfunc points to hello() */
        (*pfunc)();       /* invoke hello */
        return ;
}
```

pfunc是指向hello的函数指针，通过 (*func)() 调用

指向函数的指针

- 回调函数
  - 传递给调用者，让调用者回调
  - 传递事件处理函数给事件接收者
  - 线程结束时用回调函数释放资源
- 泛型编程与面向对象思想

```
int lower( struct student s1, struct student s2);
int higher(struct student s1, struct student s2);
void find( struct student *array, int length,
        int (*compare)(struct student s1, struct student s2));
```

compare是一个比较策略

比较策略易变，而比较算法不易变

动态内存分配

- 局部的自动变量在栈上分配，函数返回后变量销毁
- 在堆上动态的分配内存，整个进程都可见，在手动释放之前一直存在
- 程序根据需要向操作系统申请一块内存
- 必须在代码里面显式的释放，否则造成内存泄露

动态内存分配

- void * malloc(size_t size)
  - 头文件stdlib.h
  - size指定分配内在的大小，以字节为单位
  - 返回指向所分配内存的指针
  - 分配失败，返回NULL，系统已无可用内存
  - 分配成功后，转成我们希望使用的类型

```
void *ptr = malloc( sizeof(int) );
int *num = (int *)ptr;
```

分配4个字节，并把分配的内存当作int使用

动态内存分配

- void free(void *ptr);
  - stdlib.h
  - 释放先前动态分配的内存
  - 应用程序结束前用free把申请的内存还给系统

free(ptr);

指针与结构体

- struct student *p = NULL;
  - p是指针，指向 struct student类型变量
- struct class{
  char *name; /* 班级名称 */
  int count; /* 学生数 */
  };
  指针作为结构体成员
- struct class{
  char *name;
  int count;
  student *p; /* 指向学生结构体类型变量的指针作为成员
  };

指针与结构体

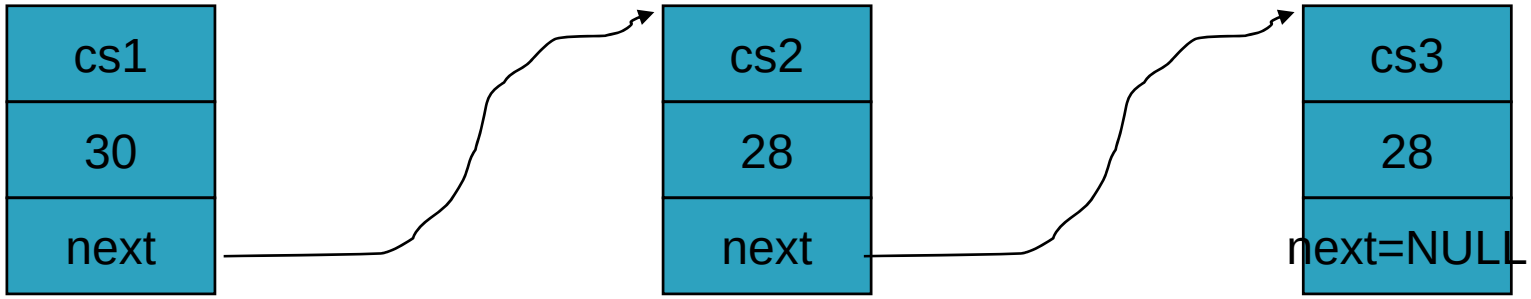
- 自引用结构，成员是指向同类型变量的指针
- struct class{

```
        char *name;             /* 班名字 */
        int count;              /* 学生数 */
        struct class *node;     /* 下一个班级 */
}
```

- struct class 类型的成员 node, 指向一个struct node 类型变量

指针与结构体

- 三个struct class 类型变量cs1 ,cs2, cs3，组成一个链状
- cs1->next = &cs2;
- cs2->next = &cs3;
- cs3->next = NULL;



指针与结构体

- 练习：动态分配一块内存，用于存储10个整形数，并依次赋值。
- 练习：动态分配3个struct student类型的变量，并串成一个链，依次打印出3个节点的成员值，程序结束前删除所有分配的内存

函数指针与结构体

```
int average_exam();          /* 考试平均分 */
int average_with_hw();       /* 加上作业后的平均分 */
struct class
{
        struct student s[30];
        int (*average)();          /* 函数指针 */
}classA, classB;
classA.average = average_exam;
classB.average = average_with_hw;
```

函数指针与结构体

- 函数指针作为结构体的一个成员
- 数据类型和相关的操作绑定在一起
- 功能类似c++多态中的虚函数
- 不同的变量可以使用不同的函数
- 对外提供一个接口
- 实现时，不同的个体可以采用具体的方法
- Linux文件系统对外提供接口，ext, fat, ntfs等具体文件系统对就的结构体变量使用不同的函数

函数指针与结构体

- 练习：出行可以选择不同的交通方式：公交车，出租车，地铁，他们都有交通工具的属性，但却使用不同的计费方式。张三从五道口去市郊，先坐乘铁到军博，后打车到长途公交车站，然后坐公交车，三种交通方式都用到了。请编程计算他一共花了多少钱。
- 三种工具的共同属性：行驶速度，乘坐时间
- 地铁时速30, 公车10, 出租车 40
- 公交车计费方式：10公里以内2元，超过10公里固定4元
- 地铁计费方式：固定2元
- 出租车：3公里内10元，每多出一公里加2元

指针的数据类型

定义	含义
int i;	定义整型变量i
int *p;	p为指向整型数据的指针变量
int a[n];	定义含n个元素的整型数组a
int *p[n];	n个指向整型数据的指针变量组成的指针数组p
int (*p)[n];	p为指向含n个元素的一维整型数组的指针变量
int f();	f为返回整型数的函数
int *p();	p为返回指针的函数，该指针指向一个整型数据
int (*p)();	p为指向函数的指针变量，该函数返回整型数
int **p;	p为指针变量，它指向一个指向整型数据的指针变量

例　下列定义的含义

（1）int *p[3]; ← 指针数组

（2）int (*p)[3]; ← 指向一维数组的指针

（3）int *p(int); ← 返回指针的函数

（4）int (*p)(int); ← 指向函数的指针，函数返回int型变量

（5）int *(*p)(int); ← 指向函数的指针，函数返回int 型指针

（6）int (*p[3])(int); ← 函数指针数组，函数返回int型变量

（7）int *(p[3])(int); ← 函数指针数组，函数返回int型指针

其他问题？