

C++ Program Design

-- C to CPP, **Pointers to Functions**

Junjie Cao @ DLUT

Summer 2021

<https://github.com/jjcao-school/c>

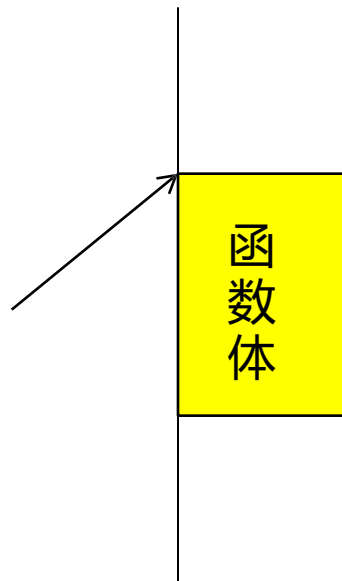
From 程序设计实习 by 郭炜 + 刘家瑛 @北京大学

函数指针

Pointers to Functions

基本概念

- 程序运行期间，每个函数都会占用一段连续的内存空间。
- 而**函数名**就是该函数所占内存区域的**起始地址**(也称“入口地址”)。
- 我们可以将函数的入口地址赋给一个**指针变量**，使该指针变量指向该函数。
- 然后通过指针变量就可以调用这个函数
- 这种指向函数的指针变量称为“**函数指针**”



定义形式

类型名 (* 指针变量名)(参数类型1, 参数类型2,...);

定义形式

类型名 (* 指针变量名)(参数类型1, 参数类型2,...);

例如：

int (*pf)(int ,char);

定义形式

类型名 (* 指针变量名)(参数类型1, 参数类型2,...);

例如：

int (*pf)(int ,char);

表示pf是一个函数指针，它所指向的函数，返回值类型应是int，该函数应有两个参数，第一个是int 类型，第二个是char类型。

使用方法

可以用一个原型匹配的函数的名字给一个函数指针赋值。

要通过函数指针调用它所指向的函数，写法为：

函数指针名(实参表);

使用方法

```
#include <stdio.h>

void PrintMin(int a,int b) {
    if( a<b )
        printf("%d",a);
    else
        printf("%d",b);
}

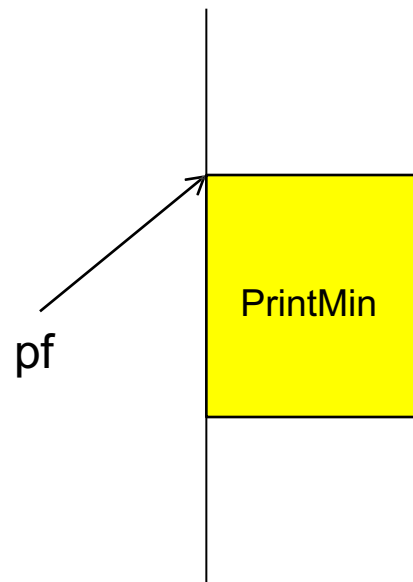
int main() {
    void (* pf)(int ,int);
    int x = 4, y = 5;
    pf = PrintMin;
    pf(x,y);
    return 0;
}
```


使用方法

```
#include <stdio.h>

void PrintMin(int a,int b) {
    if( a<b )
        printf("%d",a);
    else
        printf("%d",b);
}

int main() {
    void (* pf)(int ,int);
    int x = 4, y = 5;
    pf = PrintMin;
    pf(x,y);
    return 0;
}
```

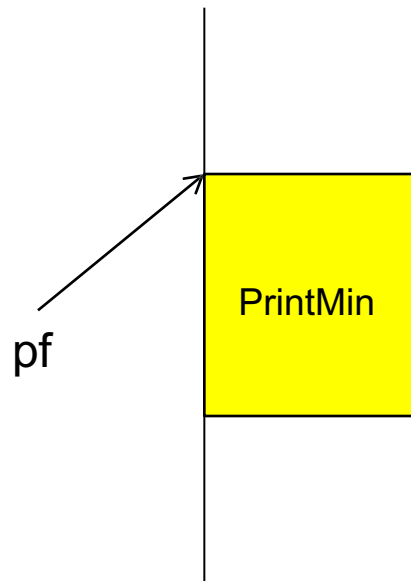


使用方法

```
#include <stdio.h>

void PrintMin(int a,int b) {
    if( a<b )
        printf("%d",a);
    else
        printf("%d",b);
}

int main() {
    void (* pf)(int ,int);
    int x = 4, y = 5;
    pf = PrintMin;
    pf(x,y);
    return 0;
}
```



输出结果：
4

函数指针和qsort库函数

C语言快速排序库函数：

```
void qsort(void *base, int nelem, unsigned int width,  
           int ( * pfCompare)( const void *, const void *));
```

可以对任意类型的数组进行排序

函数指针和qsort库函数

a[0]	a[1]	a[i]	a[n-1]
------	------	-------	------	-------	--------

对数组排序，需要知道：

函数指针和qsort库函数

a[0]	a[1]	a[i]	a[n-1]
------	------	-------	------	-------	--------

对数组排序，需要知道：

1) 数组起始地址

函数指针和qsort库函数

a[0]	a[1]	a[i]	a[n-1]
------	------	-------	------	-------	--------

对数组排序，需要知道：

- 1) 数组起始地址
- 2) 数组元素的个数

函数指针和qsort库函数

a[0]	a[1]	a[i]	a[n-1]
------	------	-------	------	-------	--------

对数组排序，需要知道：

- 1) 数组起始地址
- 2) 数组元素的个数
- 3) 每个元素的大小（由此可以算出每个元素的地址）

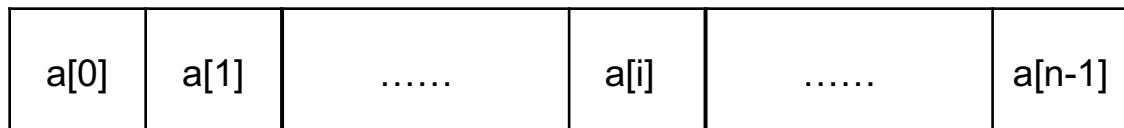
函数指针和qsort库函数

a[0]	a[1]	a[i]	a[n-1]
------	------	-------	------	-------	--------

对数组排序，需要知道：

- 1) 数组起始地址
- 2) 数组元素的个数
- 3) 每个元素的大小（由此可以算出每个元素的地址）
- 4) 元素谁在前谁在后的规则

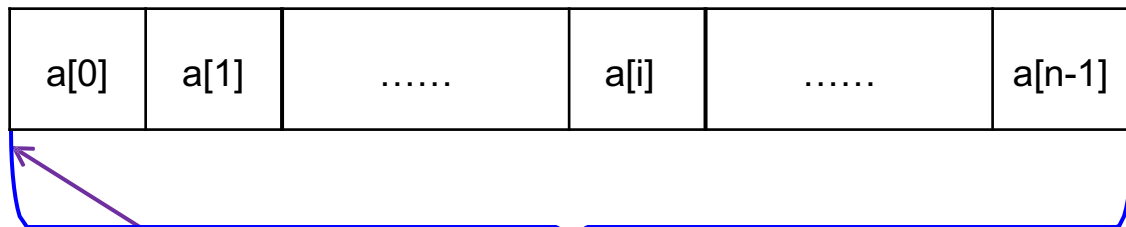
函数指针和qsort库函数



```
void qsort(void *base, int nelem, unsigned int width,  
int ( * pfCompare)( const void *, const void *));
```

base: 待排序数组的起始地址 ,

函数指针和qsort库函数

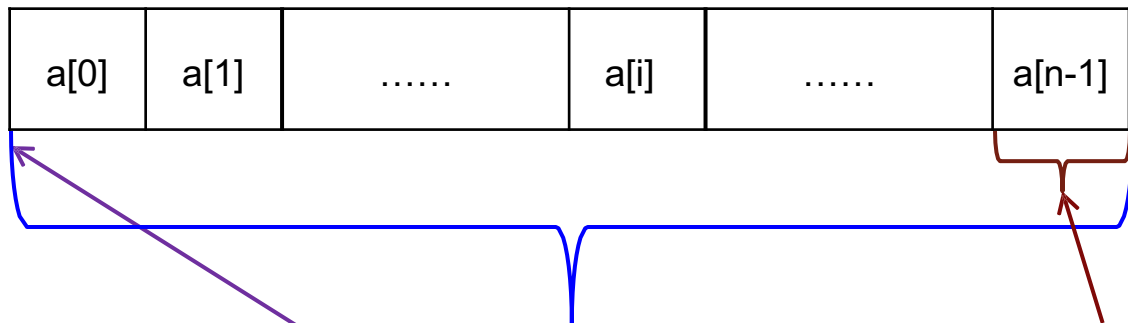


```
void qsort(void *base, int nelem, unsigned int width,  
int ( * pfCompare)( const void *, const void *));
```

`base`: 待排序数组的起始地址 ,

`nelem`: 待排序数组的元素个数 ,

函数指针和qsort库函数



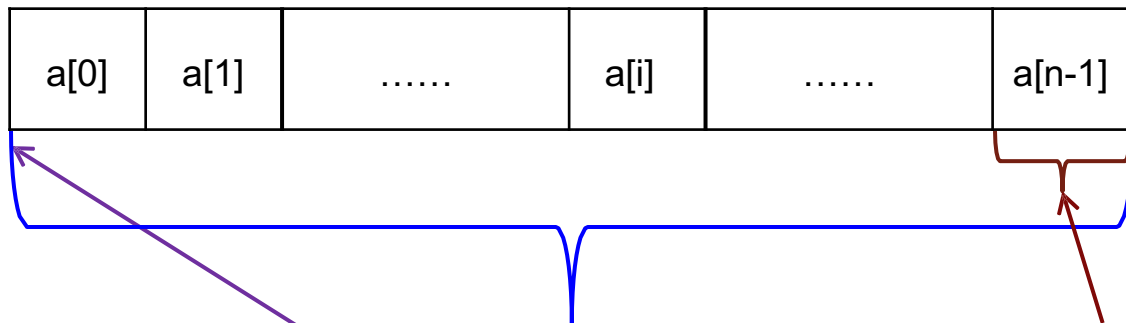
```
void qsort(void *base, int nelem, unsigned int width,  
int ( * pfCompare)( const void *, const void *));
```

base: 待排序数组的起始地址 ,

nelem: 待排序数组的元素个数 ,

width: 待排序数组的每个元素的大小 (以字节为单位)

函数指针和qsort库函数



```
void qsort(void *base, int nelem, unsigned int width,  
int ( * pfCompare)( const void *, const void *));
```

base: 待排序数组的起始地址 ,

nelem: 待排序数组的元素个数 , **width**: 待排序数组的每个元素的大小 (以字节为单位) **pfCompare**: 比较函数的地址

函数指针和qsort库函数

```
void qsort(void *base, int nelem, unsigned int width,  
    int ( * pfCompare)( const void *, const void *));
```

pfCompare: 函数指针，它指向一个“比较函数”。
该比较函数应为以下形式：

```
int 函数名(const void * elem1, const void * elem2);
```

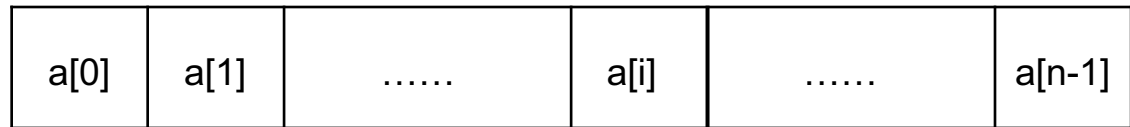
比较函数是程序员自己编写的

函数指针和qsort库函数


排序就是一个不断比较并交换位置的过程。

qsort函数在执行期间，会通过pfCompare指针调用“比较函数”，调用时将要比较的两个元素的地址传给“比较函数”，然后根据“比较函数”返回值判断两个元素哪个更应该排在前面。

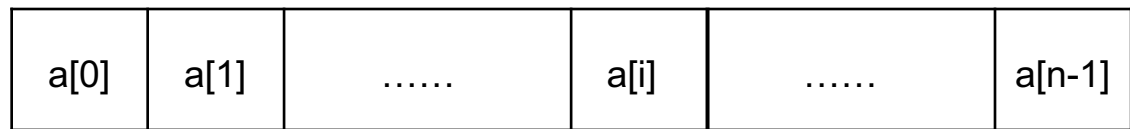
函数指针和qsort库函数



pfCompare(e1, e2);



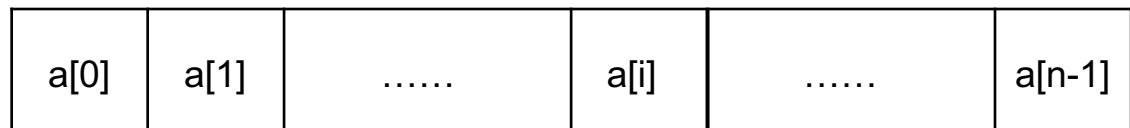
函数指针和qsort库函数



pfCompare(e1, e2);

int 比较函数名(const void * elem1, const void * elem2);

函数指针和qsort库函数



pfCompare(e1, e2);

int 比较函数名(const void * elem1, const void * elem2);

比较函数编写规则：

- 1) 如果 * elem1应该排在 * elem2前面，则函数返回值是负整数
- 2) 如果 * elem1和* elem2哪个排在前面都行，那么函数返回0
- 3) 如果 * elem1应该排在 * elem2后面，则函数返回值是正整数

函数指针和qsort库函数

实例：

下面的程序，功能是调用qsort库函数，将一个unsigned int数组按照个位数从小到大进行排序。比如 8，23，15三个数，按个位数从小到大排序，就应该是 23，15，8

```
#include <stdio.h>
#include <stdlib.h>
int MyCompare( const void * elem1, const void * elem2 )
{
    unsigned int * p1, * p2;
    p1 = (unsigned int *) elem1; // “ * elem1” 非法
    p2 = (unsigned int *) elem2; // “ * elem2” 非法
    return (* p1 % 10) - (* p2 % 10 );
}
#define NUM 5
int main()
{
    unsigned int an[NUM] = { 8,123,11,10,4 };
    qsort( an,NUM,sizeof(unsigned int), MyCompare);
    for( int i = 0;i < NUM; i ++ )
        printf("%d ",an[i]);
    return 0;
}
```

输出结果：
10 11 123 4 8