西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C++程序设计
Programming in C++

1011018

主讲：魏英，计算机学院

指针的定义与使用

- 3、指针的有效性
- 4、指针的运算

- 指针指向一个有确定存储空间的对象（称为已知对象），则该指针是有效的。

- 若一个指针不指向程序中任何已知对象，称其指向未知对象。未知对象的指针是无效的，无效的指针使用间接引用运算几乎总会导致崩溃性的异常错误。

- （1）如果指针的值为0，称为0值指针，又称空指针（null pointer），空指针是无效的。

```
int *p=0;
*p=2; //空指针间接引用将导致程序产生严重的异常错误
```

- （2）如果指针未经初始化，或者没有赋值，或者指针运算后指向未知对象，那么该指针是无效的。

```
int *p;
*p=100;     //错误，p为无效指针，不能间接引用
```

- （3）一个指针曾经指向一个已知对象，在对象的内存空间释放后，虽然该指针仍是原来的内存地址，但指针所指已是未知对象，称为“迷途指针”（dangling pointer）。

```
char *p=NULL; //p是空指针，全局变量
void fun()
{
    char c; //局部变量
    p = &c; //指向局部变量c，函数调用结束后，c的空间释放，p就成了迷途指针
}
void caller()
{
    fun();
    *p=2; //p现在是迷途指针
}
```

- 指针的运算都是作用在连续存储空间上才有意义。
- （1）指针加减整数运算

```
int x[10], n=3 , *p=&x[5];
p+1 //指向存储空间中x[5]后面的第1个int型存储单元
p+n //指向存储空间中x[5]后面的第n(3)个int型存储单元
p-1 //指向存储空间中x[5]前面的第1个int型存储单元
p-n //指向存储空间中x[5]前面的第n(3)个int型存储单元
```

- （2）指针变量自增自减运算
- 设p是一个指针变量，其自增自减运算包括p++、++p、p--、--p形式。

```
int x[10], *p=&x[5];
p++ //p指向x[5]后面的第1个int型内存单元
++p //p指向x[5]后面的第1个int型内存单元
p-- //p指向x[5]前面的第1个int型内存单元
--p //p指向x[5]前面的第1个int型内存单元
```

- （3）两个指针相减运算
- 设p1、p2是相同类型的两个指针（常量或变量），则p2-p1的结果为两个指针之间对象的个数，如果p2的地址值大于p1结果为正，否则为负。

```
int x[5], *p1=&x[0],*p2=&x[4];
int n;
n=p2-p1;//n的值为4
```

- （4）指针的关系运算
- 设p1、p2是同一个指向类型的两个指针（常量或变量），则p2和p1可以进行关系运算，用于比较这两个地址的位置关系。

```
int x[5], *p1=&x[0],*p2=&x[4];
p2>p1 //表达式值为真
```

- 指针的const限定
- （1）一个指针变量可以指向只读型对象，称为指向const对象的指针，定义形式为：

```
const 指向类型 *指针变量,.....;
```

- 即在指针变量定义前加const限定符，其含义是不允许通过指针来改变所指向的const对象的值。

```
const int a=10, b=20;
const int *p;
p=&a; //正确，p不是只读的
p=&b; //正确，p不是只读的
*p = 42; //错误，*p是只读的
```

- 把一个const对象的地址赋给一个非const对象的指针变量是错误的，例如：

```
const double pi = 3.14;
double *ptr = &pi; //错误，ptr是非const指针变量
const double *cptr=&pi;//正确，cptr是const指针变量
```

- 允许把非const对象的地址赋给指向const对象的指针；不能使用指向const对象的指针修改指向对象，然而如果该指针指向的是一个非const对象，可用其他方法修改其所指的对象。例如：

```
const double pi = 3.14;
const double *cptrf = &pi; //正确
double f = 3.14; //f是double型，f是非const
cptrf = &f; //正确，允许将f的地址赋给cptrf
f=1.618; //正确，可以修改f的值
*cptrf = 10.1; //错误，不允许通过引用cptrf修改f的值
```

- 实际编程中，指向const的指针常用作函数的形参，以此确保传递给函数的实参对象在函数中不被修改。

```
void fun( const int *p)
{
      ……
}

int main()
{  int a;
   fun(&a);
}
```

- （2）const指针
- 一个指针变量可以是只读的，称为const指针，定义形式为：

```
指向类型 * const 指针变量,.....;
```

```
int a=10, b=20;
int *const pc = &a; //pc是const指针
pc = &b; //错误，pc是只读的
pc = pc; //错误，pc是只读的
pc++; //错误，pc是只读的
*pc=100; //正确，a被修改
```

- pc是指向int型对象的const指针。
- 不能使pc再被赋值指向其他对象。任何企图给const指针赋值的操作都会导致编译错误。
- 但可通过pc间接引用修改该对象的值。

- （3）指向const对象的const指针
- 可以定义指向const对象的const指针，形式为：

```
const 指向类型 * const 指针变量,.....;
```

```
const double pi = 3.14159;
const double * const cpc = &pi;
 //cpc为指向const对象的const指针
```

C程序设计