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

# 动态分配内存

1、new与delete运算

2、动态内存的应用

- 在使用数组的时候，总有一个问题困扰着我们：
- 数组应该有多大？

- 而如下操作是错误的：

```
int n;
cin>>n; //输入n值
double A[n][n]; //意图由输入n值确定数组大小
```

- C++内存分配有两种方式：静态分配和动态分配。

- 静态分配指在编译时为程序中的数据对象分配相应的存储空间，本书前面所有例子中的变量、数组、指针定义等均是静态分配方式。
- 由于是在编译时为数据对象分配存储空间，因此就要求在编译时空间大小必须是明确的，所以数组的长度必须是常量。而一旦编译完成，运行期间这个数组的长度就是固定不变的。

- 动态分配是程序运行期间根据实际需要动态地申请或释放内存的方式，它不象数组等静态内存分配方式那样需要预先分配存储空间，而是根据程序的需要适时分配，且分配的大小就是程序要求的大小。因此，动态分配方式有如下特点：

- ①不需要预先分配存储空间；
- ②分配的空间可以根据程序的需要扩大或缩小；

- new与delete运算

表20-1 new和delete运算符

| 运算符 | 功能 | 目 | 结合性 | 用法 |
| --- | --- | --- | --- | --- |
| new | 动态分配 | 单目 | 自右向左 | new type |
| new [] | 动态分配数组 | 单目 | 自右向左 | new type [] |
| delete | 释放空间 | 单目 | 自右向左 | delete expr |
| delete [] | 释放数组空间 | 单目 | 自右向左 | delete [] expr |

```
int *p1, *p2;
char *pz1,*pz2;
p1=new int; //分配一个整型空间，若成功则p1指向该空间，否则p1为NULL
p2=new int(10); //分配一个整型空间，且给这个整型赋初值10，即*p2为10
pz1=new char[80]; //分配一个字符数组（字符串）空间，即pz1为字符串指针
pz2=new char[5][80]; //分配一个二维字符数组（字符串数组）空间，即pz1为字符串数组指针
delete p1; //释放p1指向的整型空间
delete [] pz1; //释放pz1指向的字符串空间
```

- （1）new运算结果是指向分配得到的内存空间的指针，如果没有足够的内存空间可以分配，其运算结果是一个0值指针。

- （2）销毁对象后，指针p1变成没有定义，然而它仍然存放先前所指向的对象（已销毁）的地址，因此指针p1不再有效，称这样的指针为迷途指针。通常在delete运算之后将指针重设为0值指针，避免迷途指针。

- （3）用new创建的动态对象使用完后，必须用delete销毁它。
- （4）delete只能删除由new创建的动态对象，否则将导致程序错误。

- （1）静态内存管理由编译器进行，程序员只做对象定义（相当于分配），而动态内存管理按程序员人为的指令进行。

- （2）动态内存分配和释放必须对应，即有分配就必须有释放，不释放内存会产生“内存泄漏”，后果是随着程序运行多次，可以使用的内存空间越来越少；另一方面，再次释放已经释放的内存空间，会导致程序出现崩溃性错误。

- （3）静态分配内存的生命期由编译器自动确定，要么是程序运行期，要么是函数执行期。动态分配内存的生命期由程序员决定，即从分配时开始，至释放时结束。特别地，动态分配内存的生命期允许跨多个函数。

- （4）静态分配内存的对象有初始化，动态分配内存一般需要人为的指令赋初始值。

- （5）避免释放内存后出现“迷途指针”，应及时设置为空指针。

【例20.1】动态内存举例。

```
#include <iostream>
using namespace std;
int *f1(int n) //分配n个整型内存，返回首地址
{ int *p, i;
  p = new int[n]; //分配
  for (i=0; i<n; i++) p[i]=i; //赋初始值
  return p; //动态分配的指针返回是有意义的
}
void f2(int *p,int n) //输出动态内存中的n个数据
{ while (n-->0) cout<<*p++<<" "; }
void f3(int *p)
{ delete [] p; } //释放内存
```

```
13 int main()
14 {
15    int *pi;
16    pi=f1(5); //分配
17    f2(pi,5); //输出
18    f3(pi); //释放
19    return 0;
20 }
```

# C程序设计