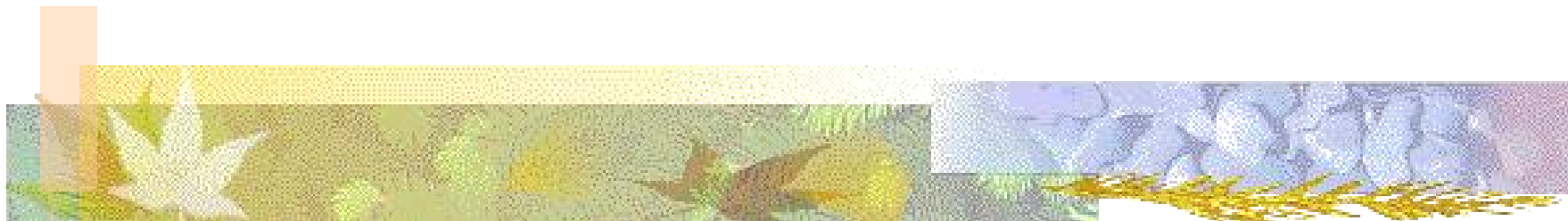


规格严格 功夫到家



动态数组




哈尔滨工业大学（深圳）

计算机科学与技术学院

陈清财

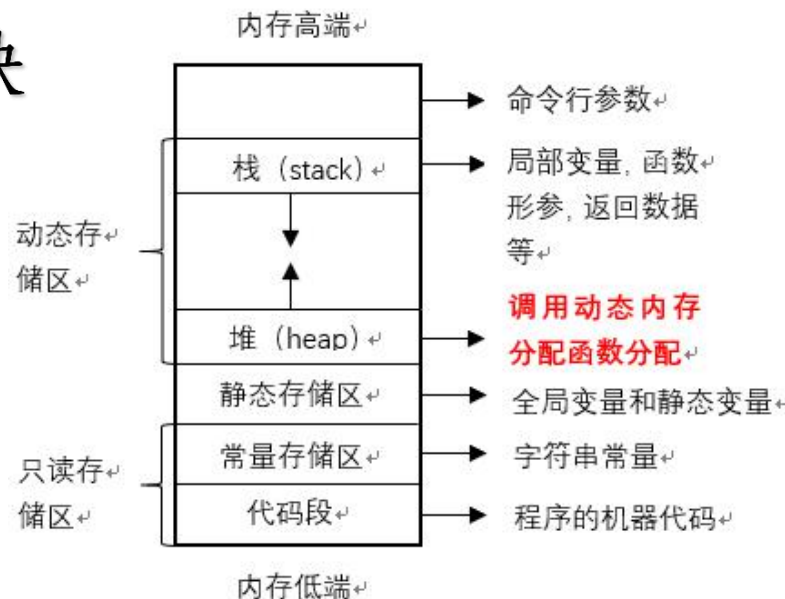


- 
- 定长数组存在的问题：
 - 1. 空间效率差（固定长度）
 - 2. 容易出错（溢出）

11.4 动态数组

11.4.1 C程序的内存映像

■ 编译后的C程序获得并使用4块在逻辑上不同且用于不同目的的内存存储区



— 只读存储区

- 存放程序的机器代码和字符串常量等只读数据。

— 从静态存储区分配

- 全局变量和静态变量

11.4 动态数组

11.4.1 C程序的内存映像

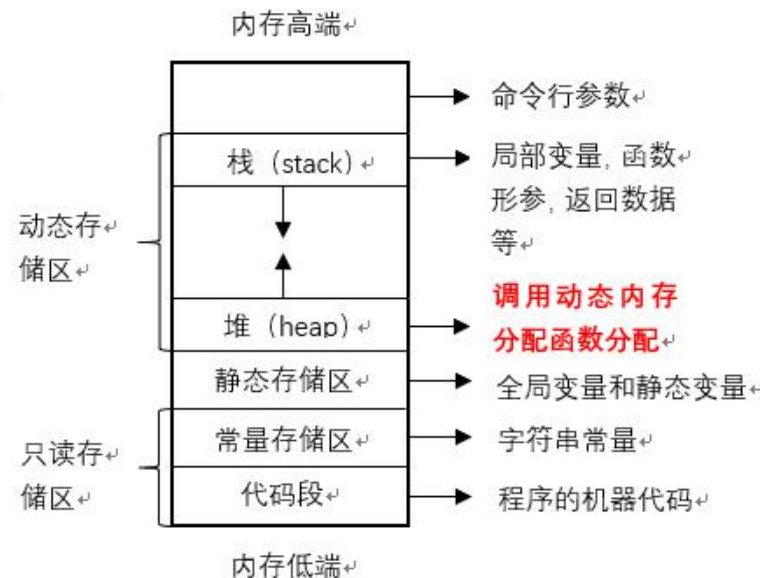
■ 编译后的C程序获得并使用4块在逻辑上不同且用于不同目的的内存存储区

— 在栈上创建

- 存放函数参数值、局部变量值等
- 在执行函数调用时，系统在栈上为函数内的局部变量及形参分配内存，函数执行结束时，自动释放这些内存

— 从堆上分配

- 在程序运行期间，用动态内存分配函数来申请的内存都是从堆上分配的，动态内存的生存期由程序员自己来决定

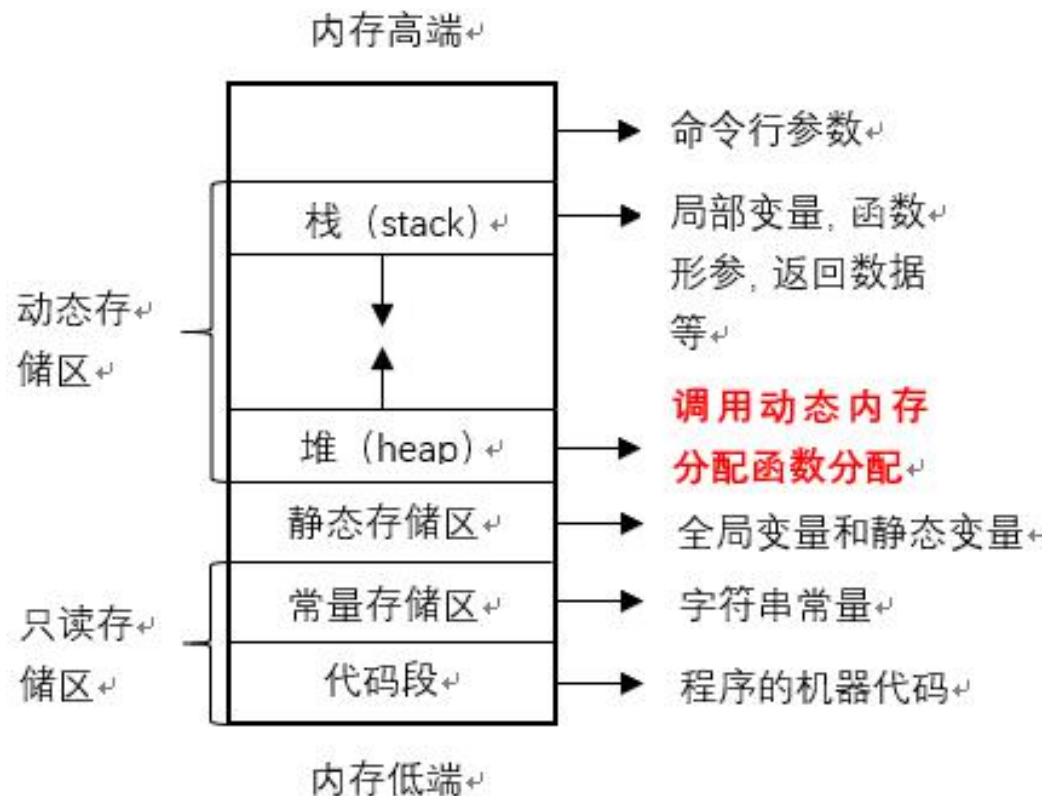


11.4 动态数组

11.4.1 C程序的内存映像

■ 堆和栈的扩展方向

■ 内存泄漏的概念



11.4.2 动态内存分配函数

- Two primary methods of allocating memory:

```
#include <stdlib.h>
#include <alloc.h>
```

```
void* malloc(unsigned int size);
```

```
void* calloc(unsigned int num,
              unsigned int size);
```

void*类型的指针可以指向任意类型的变量，通常强转 (**Type***) 为其他类型

11.4.2 动态内存分配函数

- Two primary methods of allocating memory:

```
void* malloc(unsigned int size);
```

- 向系统申请大小为**size**的内存块
- 把首地址返回，若申请不成功则返回**NULL**

```
void* calloc(unsigned int num,  
             unsigned int size);
```

- 向系统申请**num**个**size**大小的内存块
- 把首地址返回，若申请不成功则返回**NULL**

11.4.2 动态内存分配函数

- Two primary methods of allocating memory:

```
void* malloc(unsigned int size);
```

例如: `int *pi = NULL;`
`pi = (int *)malloc(sizeof(int));`

```
void* calloc(unsigned int num,  
             unsigned int size);
```

例如: `float *pf = NULL;`
`pf = (float *)calloc(10, sizeof(float));`

11.4.2 动态内存分配函数

■ Method of deallocating memory:

```
void free(void* p) ;
```

- 释放由malloc() 和calloc() 申请的内存块
- p是指向此块内存的指针
- free时系统标记此块内存为未占用，可被重新分配

11.4.2 动态内存分配函数

■ Method of reset allocation:

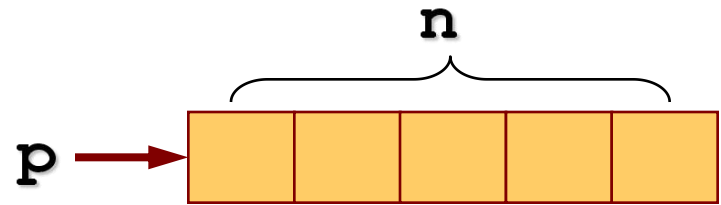
```
void* realloc(void* p , unsigned int size) ;
```

- 改变原来分配的存储空间的大小
- **p**是指向此块内存的指针，**size**是新内存块的大小
- 函数返回新分配存储空间首地址，与原来分配的首地址不一定相同

11.4.3 长度可变的一维动态数组

【例11.6】 一维动态数组

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  void InputArray(int *p, int n);
4  double Average(int *p, int n);
5  int main()
6  {
7      int *p = NULL, n;
8      double aver;
9      printf("How many students?");
10     scanf("%d", &n);
11     p = (int *) malloc(n * sizeof(int));
12     if (p == NULL)
13     {
14         printf("No enough memory!\n");
15         exit(1);
16     }
17     printf("Input %d score:", n);
18     InputArray(p, n);
19     aver = Average(p, n);
20     printf("aver = %.1f\n", aver);
21     free(p);
22     return 0;
23 }
```



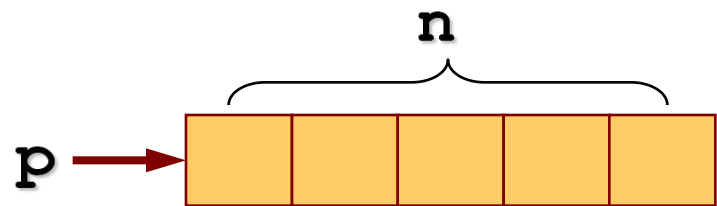
确保指针使用前是非空指针

释放向系统申请的存储空间

11.4.3 长度可变的一维动态数组

【例11.6】 一维动态数组

```
24  /* 形参声明为指针变量，输入数组元素值 */
25  void InputArray(int *p, int n)
26  {
27      int i;
28      for (i=0; i<n; i++)
29      {
30          scanf("%d", &p[i]);
31      }
32  }
33  /* 形参声明为指针变量，计算数组元素的平均值 */
34  double Average(int *p, int n)
35  {
36      int i, sum = 0;
37      for (i=0; i<n; i++)
38      {
39          sum = sum + p[i];
40      }
41      return (double)sum / n;
42  }
```



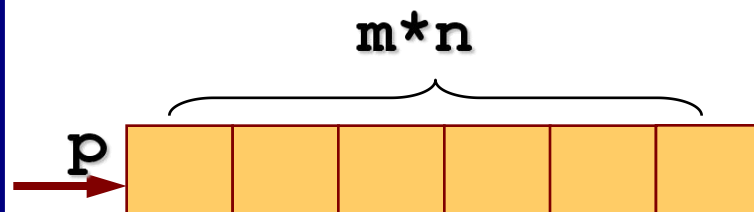
像使用一维数组一样
使用动态数组

How many students? 5✓

Input 5 score: 90 85 70 95 80✓

aver = 84.0


```
1  #include <stdio.h>
2  #include <stdlib.h>
3  void InputArray(int *p, int m, int n);
4  double Average(int *p, int m, int n);
5  int main()
6  {
7      int *p = NULL, m, n;
8      double aver;
9      printf("How many classes?");
10     scanf("%d", &m);
11     printf("How many students in a class?");
12     scanf("%d", &n);
13     p = (int *)calloc(m*n, sizeof(int));
14     if (p == NULL)
15     {
16         printf("No enough memory!\n");
17         exit(1);
18     }
19     InputArray(p, m, n);
20     aver = Average(p, m, n);
21     printf("aver = %.1f\n", aver);
22     free(p);
23     return 0;
24 }
```



确保指针使用前是
非空指针

释放向系统申请的
存储空间

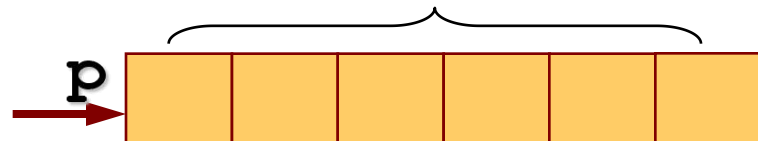
11.4.4

【例11.7】

二维动态数组

仍当做一维数组
来使用

$m*n$



```
25  /* 形参声明为指向二维数组的列指针，输入数组元素值 */
26  void InputArray(int *p, int m, int n)
27  {
28      int i, j;
29      for(i = 0; i<m; i++)          /* m 个班 */
30      {
31          printf("Please enter scores of class %d:\n", i+1);
32          for(j = 0; j<n; j++)      /* 每班 n 个学生 */
33          {
34              scanf("%d", &p[i*n+j]);
35          }
36      }
37  }
```

```
38  /* 形参声明为指针变量，计算数组元素的平均值 */
39  double Average(int *p, int m, int n)
40  {
41      int i, j, sum = 0;
42      for(i = 0; i<m; i++)          /* m 个班 */
43      {
44          for(j = 0; j<n; j++)      /* 每班 n 个学生 */
45          {
46              sum = sum + p[i*n+j];
47          }
48      }
49      return (double)sum / (m*n);
50  }
```

How many classes? 3✓

How many students in a class? 4✓

Please enter scores of class 1:

81 72 73 64✓

Please enter scores of class 2:

65 86 77 88✓

Please enter scores of class 3:

91 90 85 92✓

aver = 80.3

■ Questions and answers

