清华大学
Tsinghua University

程序设计基础

Fundamental of Programming

清华大学软件学院

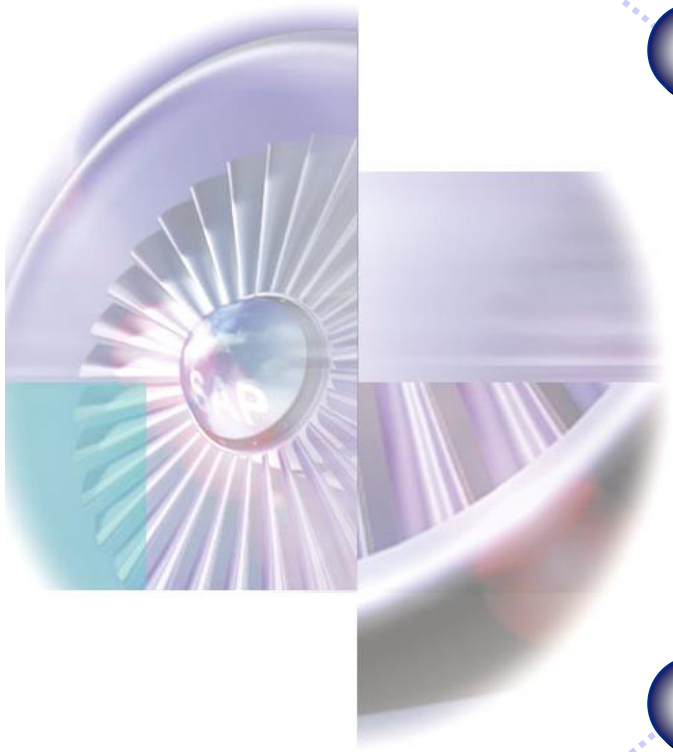
刘玉身

liuyushen@tsinghua.edu.cn

Lecture 5 : Review

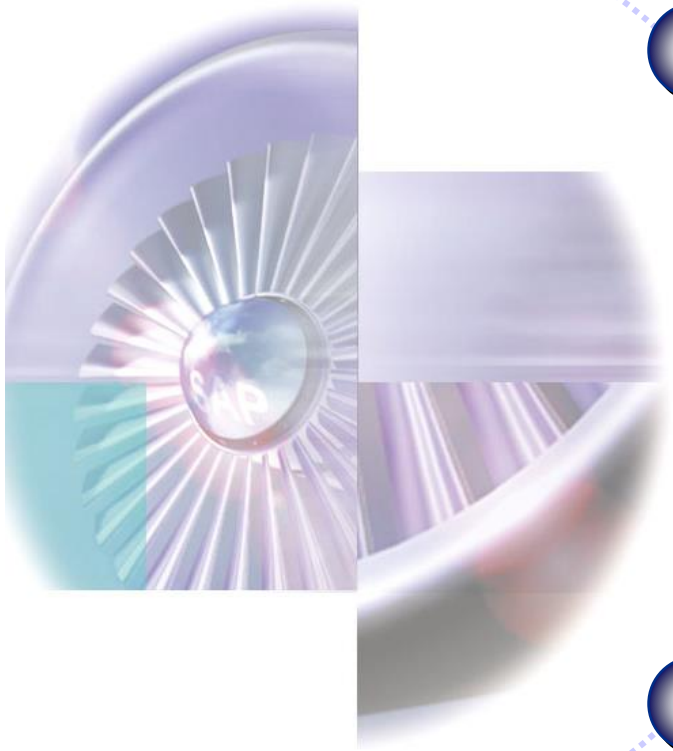
- **Arrays（数组）**
  - **The concept of array**
  - **Defining arrays**
  - **Initializing arrays**
  - **Character arrays**
  - **Multidimensional arrays**
  - **Variable length arrays**

Lecture 5: 数组



1. 一维数组的定义和引用
2. 二维数组的定义和引用
3. 字符数组
4. 程序举例

Lecture 5: 数组



1. 一维数组的定义和引用
2. 二维数组的定义和引用
3. 字符数组
4. 程序举例

一维数组的定义和引用

过山车有何特点？

1. 若干节车厢组成
2. 物是人非
3. 有序
4. 大小相同
5. 有名字(风神、飞天凤凰)

什么是数组？

在程序设计中，为了处理方便，把具有相同类型的若干变量按有序的形式组织起来。这些按序排列的同类数据元素的集合称为数组。

定义：一组有序、有名、具有相同数据类型的变量。

- 有名：数组有一个名字，如RollerCoaster；
- 有序：数组元素连续存放，0, 1, 2, …；
- 同类型：长度相同，如int, double, char；
- 变量：数组元素的值可变。

一维数组的定义方式

指明了数组元素的个数

类型说明符　　数组名［常量表达式］；

例如：char RollerCoaster[5];

表示数组名为RollerCoaster，它有5个字符型元素。

RollerCoaster

喜羊羊	美羊羊	懒羊羊	沸羊羊	

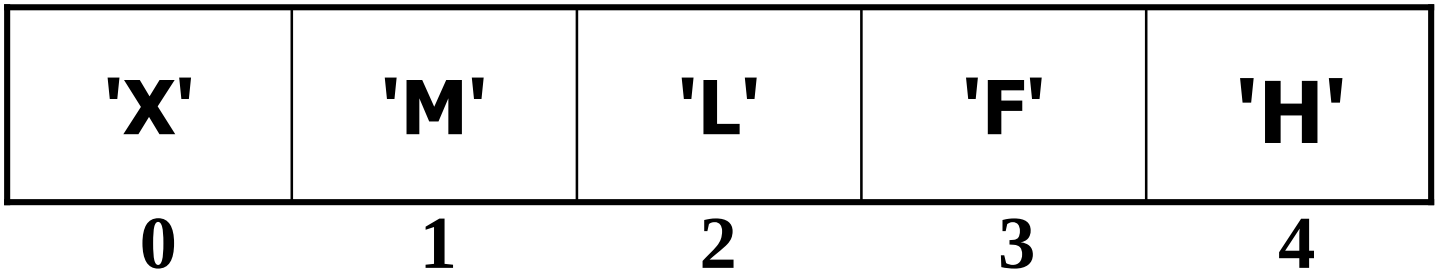
数组元素的访问

数组名[数组下标]

说明：数组下标从 0 开始

例如：RollerCoaster[4] = 'H';

RollerCoaster



数组下标的边界

- 边界：指数组下标的最小值和最大值
- 若数组长度为N，则数组下标的边界为 0～N-1
- 编译器不会进行数组下标的越界检查，一旦越界，可能会破坏程序或数据。

- int a[10]; int s = sizeof (a); // s = sizeof(int)*10
- a[0]=5; a[9]=7; //合法访问
- a[10]=3; a[-1]=6; //非法访问
- 数组长度必须是常量、符号常量，不能是变量
- int *b[3]; // ? { int*, int*, int*}, b[i] = int*;

```
#include <stdio.h>
int main()
{
    int N = 5;
    int a[N] = {0, 1, 2, 3, 4};
    return 0;
}
```

✗

```
#include <stdio.h>
#define N 5
int main()
{
    int a[N] = {0, 1, 2, 3, 4};
    return 0;
}
```

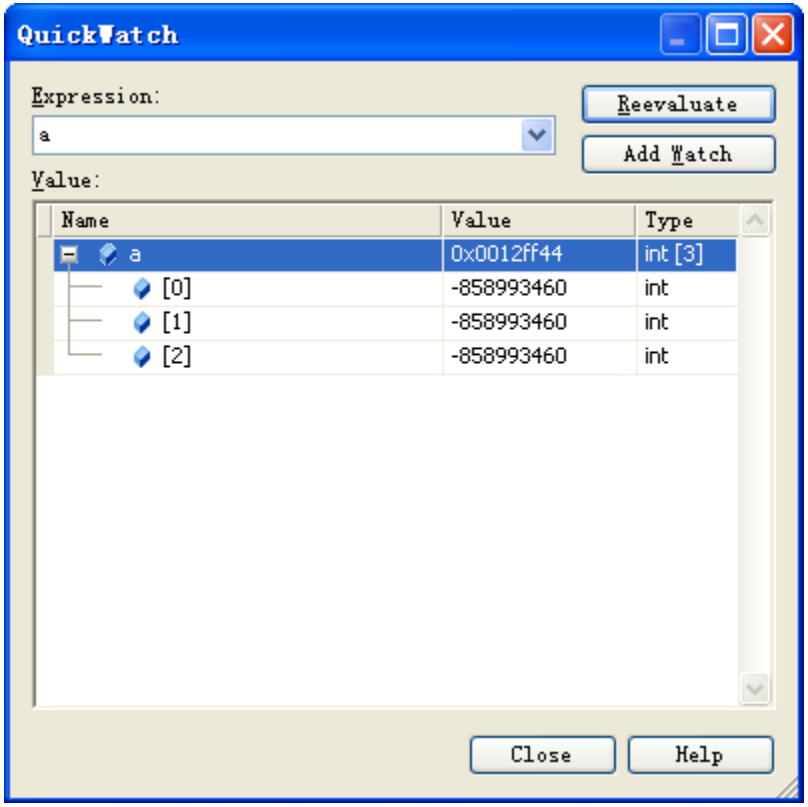
√

```
#include <stdio.h>
int main()
{
    const int N = 5;
    int a[N] = {0, 1, 2, 3, 4};
    return 0;
}
```

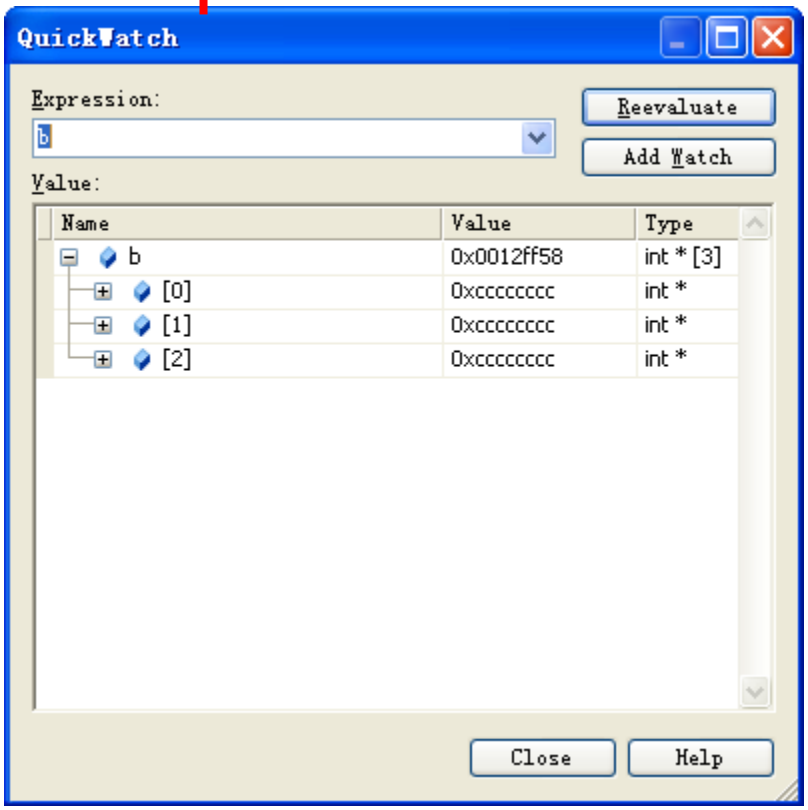
✗ (C); √ (C ++)

指针数组

- int a[3];



int *b[3];



指针数组, 就是指针的数组, 数组的元素是指针. “指针”一节会详细介绍

数组初始化

注意几种括号区别: [ ], { }, ( )

- **int arr[5] = { 0, 0, 0, 0, 0 };**
- **char letters[5] = { 'a', 'b', 'c', 'd', 'e' };**
- 初值个数小于数组元素个数，剩余的自动补0：
  float data[500] = { 100.0, 300.0, 500.5 };
  int a[500] = { 0 }; // 数组初始化为0
- **int arr[ ] = { 1, 2, 3, 4, 5 };** 等价于
  int arr[5] = { 1, 2, 3, 4, 5 };
- **char b[ ] = "Hello"; // sizeof(b)等于多少？**

分析运行结果

```
#include <stdio.h>
int main()
{
        int a[] = {10, 3, 5, 7, 2, 9, 3, 5, 7, 1};
        int i, j, N;
        N = sizeof(a)/sizeof(int);
        for(i = 0; i < N; i++)
        {
            for(j = 0; j < a[i]; j++)
                printf("*");
            printf("\n");
        }
        return 0;
}
```

```
**********

***

*****

*******

**

*********

***

*****

*******

*
```

常见错误示例

```
int a;
float a[10];  // 重复定义
```

```
int a[ ];       // 没有定义数组长度
```

```
int N;
scanf("%d", &N);
int a[N];       // 变量n不能在声明数组中使用
```

```
for(i=0; i<10; i++)
    printf("%d", a); // 只能打印a[i]
```

分析代码

问题描述：

某电视台正在举办“超级选秀比赛”，共有10 位选手入围了最后的决赛。根据比赛规则，由观众通过手机短信的方式来给选手投票，得票最多的选手获胜。请编写一个投票程序，帮助组委会统计出各个选手的得票情况。

```
#include <stdio.h>
int main( )
{
    int votes[] = {
          1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
          1, 6, 3, 8, 6, 10, 3, 8, 2, 7,
          6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
          5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
    int i, singers[11] = {0};

    int N = sizeof(votes)/sizeof(int);

    for(i = 0; i < N; i++)
    {
        singers[votes[i]]++;
    }
    for(i = 1; i <= 10; i++)
        printf("%d:\t %d\n", i, singers[i]);
    return 0;
}
```

```
1:    2
2:    2
3:    2
4:    2
5:    5
6:    11
7:    5
8:    7
9:    1
10:   3
```

2017秋期末考题1：考试成绩可视化

一、（20 分）考试成绩可视化

每次考试过后，老师都要看下大家考的怎么样。因此，需要绘制一个分数分布直方图。

现在需要在命令行中输出这个分布图。我们将考生分数按 10 分为阶梯，分成 10 个区间段，分别是[0, 10), [10, 20), …, [80, 90), [90, 100]。其中[a,b)表示**大于等于 a，小于 b** 的区间（特别注意 100 包含在最后一个区间中）。而后，我们统计每个分数段中学生的个数，以分数段为纵轴，以每个分数段中的学生个数为横轴，就可以画出分数分布直方图了。

为了简化问题，规定分布直方图的大小为 10*10，横轴每格表示 1 人。当某个分数段有 X 人时，若 X<=10，则横轴长度为 X；若 X>10，横轴长度为 10。（注意：此处的直方图是横着画的，与常见的直方图有所不同）

现给出一次考试中 N 个学生的分数，请你画出这场考试的分数分布直方图。

输入：

第一行一个整数 N。
第二行有 N 个整数 a_i，表示每个学生的分数。
$0 < N \leq 100$ ， $0 \leq a\_i \leq 100$

输出：

输出考试分数分布直方图。直方图大小为 10*10，其中’ . ’表示空白，’@’表示非空白。

样例输入：

```
36
100 99 98 98 90 95 80 81 82 83 84 85 86 87 88 89 89 70 71 72 73 74 75 76 60 64 62
63 64 55 51 52 42 42 30 2
```

样例输出：

```
@.........
..........
..........
@.........
@@........
@@@.......
@@@@@.....
@@@@@@@...
@@@@@@@@@@
@@@@@@....
```

2017秋期末考题1：考试成绩可视化

```
#include <iostream>
#include <algorithm>
using namespace std;
int num[10];
int main(){
    int n;
    cin >> n;
    for(int i = 0; i < n; i++){
        int a_i;
        cin >> a_i;
        if(100 == a_i)
            a_i = 99;
        num[a_i/10] ++;
    }
    for(int i = 0; i < 10; i++){
        for(int j = 0; j < min(num[i], 10); j++){
            cout << "@";
        }
        for(int j = num[i]; j < 10; j++)
            cout << ".";
        cout << endl;
    }
    return 0;
}
```

```
36

100 99 98 98 90 95 80 81 82 83 84 85 86 87 88 89 89
70 71 72 73 74 75 76 60 64 62 63 64 55 51 52 42 42
30 2
```

```
@.........

..........

..........

@.........

@@........

@@@.......

@@@@@.....

@@@@@@@...

@@@@@@@@@@

@@@@@@....
```

2017秋期末考题1：考试成绩可视化

Reference: <algorithm> : min, max

Returns the smallest/largest of a and b. If both are equivalent, a is returned.

```
// min example
#include <iostream>     // std::cout
#include <algorithm>    // std::min

int main () {
  std::cout << "min(1,2)==" << std::min(1,2) << '\n';
  std::cout << "min(2,1)==" << std::min(2,1) << '\n';
  std::cout << "min('a','z')==" << std::min('a','z') << '\n';
  std::cout << "min(3.14,2.72)==" << std::min(3.14,2.72) << '\n';
  return 0;
}
```

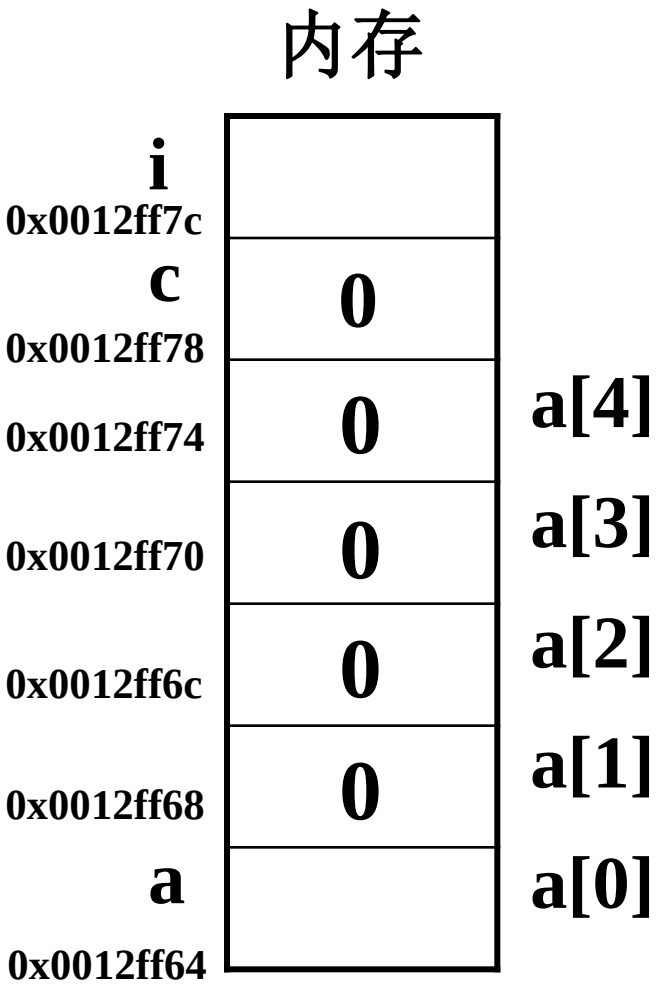
```
min(1,2)==1
min(2,1)==1
min('a','z')==a
min(3.14,2.72)==2.72
```

分析运行结果

```
int  main( )
{
      int  i, c, a[5];
      c  =  1;
      for(i  =  1;  i  <=  5;  i++)
      {
            a[i]  =  0;
      }
      printf("%d\n",  c);
      return 0;
}
```

其他顺序?

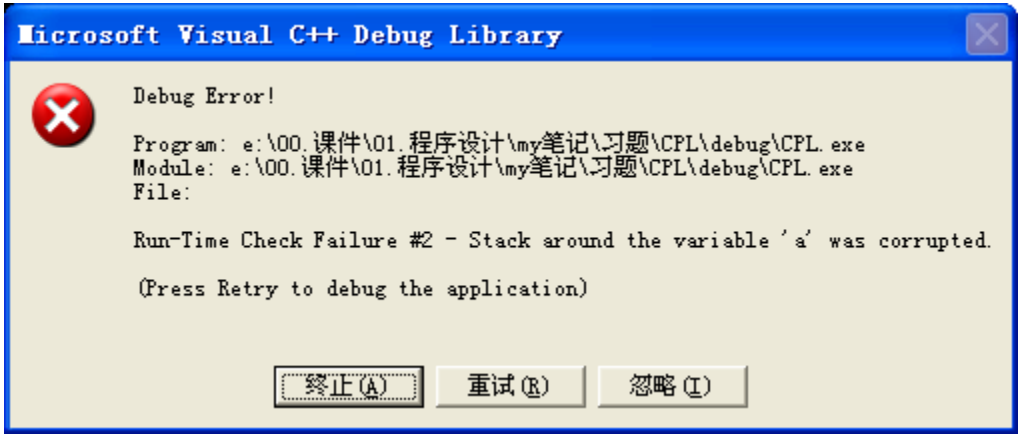
结果：c = 0;



变量在内存的分布与编译器有关(上例结果仅在VC6.0中出现)

数组访问越界的问题

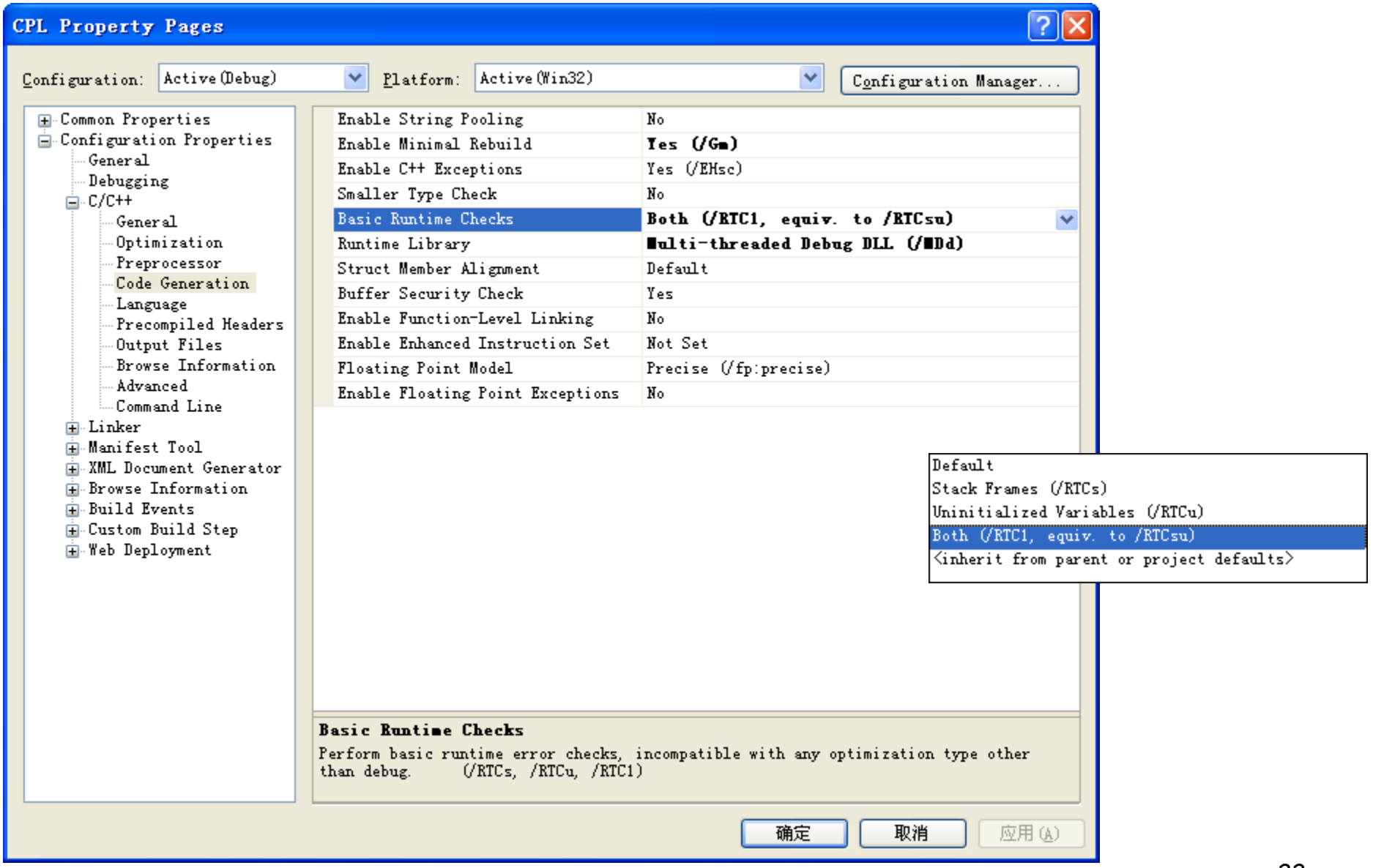
上述示例在VC2005（VC7）版本下则会报错：“Run-Time Check Failure #2 - Stack around the variable 'a' was corrupted.”

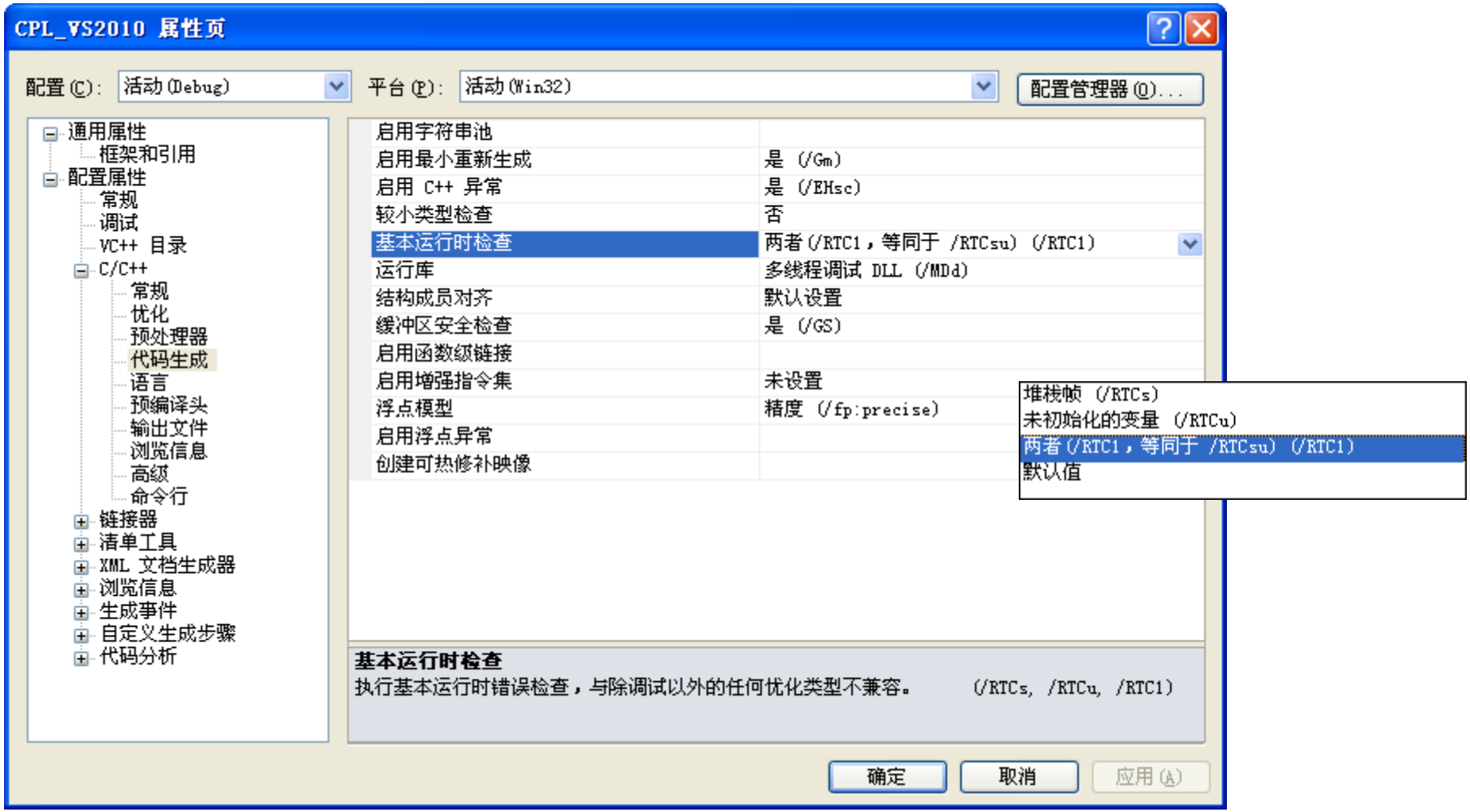


把“project->配置属性->c/c++->代码生成->基本运行时检查”设置为默认值，就没有这样的错误了。

关于MSDN的解释是在堆栈外面读写某数据，错误是名为RTC1的编译器检测的。

Run-Time Check




CPL_VS2010 属性页
配置(C): 活动(Debug)
平台(P): 活动(Win32)
配置管理器(O)...
通用属性
框架和引用
配置属性
常规
调试
VC++ 目录
C/C++
常规
优化
预处理器
代码生成
语言
预编译头
输出文件
浏览信息
高级
命令行
链接器
清单工具
XML 文档生成器
浏览信息
生成事件
自定义生成步骤
代码分析
启用字符串池
启用最小重新生成 是 (/Gm)
启用 C++ 异常 是 (/EHsc)
较小类型检查 否
基本运行时检查 两者(/RTC1，等同于 /RTCsu) (/RTC1)
运行库 多线程调试 DLL (/MDd)
结构成员对齐 默认设置
缓冲区安全检查 是 (/GS)
启用函数级链接
启用增强指令集 未设置
浮点模型 精度 (/fp:precise)
启用浮点异常
创建可热修补映像
堆栈帧 (/RTCs)
未初始化的变量 (/RTCu)
两者(/RTC1，等同于 /RTCsu) (/RTC1)
默认值
基本运行时检查
执行基本运行时错误检查，与除调试以外的任何优化类型不兼容。 (/RTCs, /RTCu, /RTC1)
确定
取消
应用(A)

前三名

问题描述：

学校要选拔三名同学参加北京市数学竞赛，为在比赛中取得好成绩，学校决定在所有学生中，挑选平时成绩排在前三名的同学组队参赛。请编写一个程序，输入所有学生的成绩，然后找出其中的前三名（学生成绩为整数，人数不确定，但不会超过200人，输入-1时结束）。

问题分析	编程模式
1. 数据个数不确定， 当输入-1时结束	循环条件永真
2. 寻找第1名	最大值/最小值
3. 寻找前3名	最大值+调整

求最大值

```
First = 0;
while(1)
{
        ……
        if(Scores[i] > First)
                First = Scores[i];
        ……
}
```

Why 0?

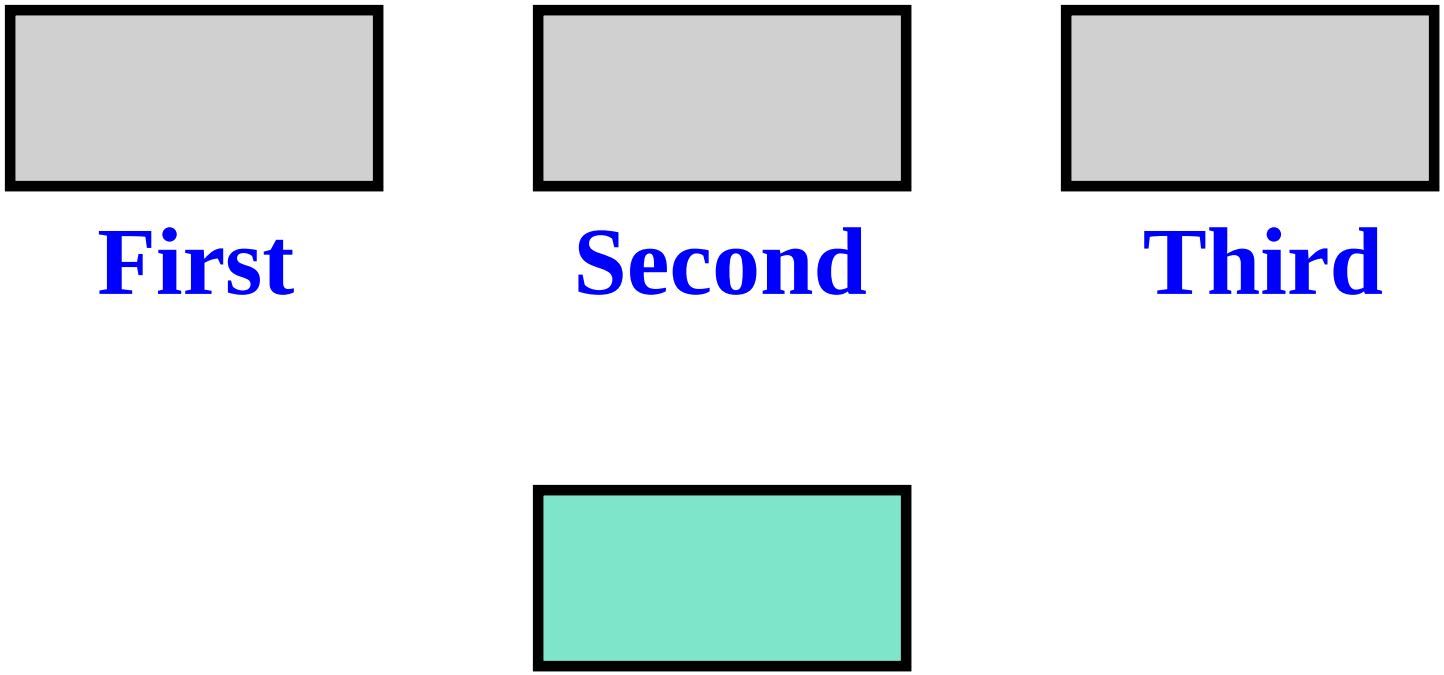
i	Scores[i]	First
初始化	—	0
0	60	60
1	70	70

求“最大值”&“最小值”模式

```
MaxScore = 0;
MinScore = 100;
while(1)
{
        ……
        if(Scores[i] > MaxScore)
                MaxScore = Scores[i];
        if(Scores[i] < MinScore)
                MinScore = Scores[i];
        ……
}
```

广泛应用于数组排序中

寻找前3名



```
int main( )
{
    int i, First, Second, Third, Scores[200];
    First = 0;
    Second = 0;
    Third = 0;
    i = 0;
    while(1)
    {
        printf("请输入第%d个学生的成绩:", i+1);
        scanf("%d", &Scores[i]);
        if(Scores[i] == -1) break;
        if(Scores[i] > First)
        {
            Third = Second;
            Second = First;
            First = Scores[i];
        }
```

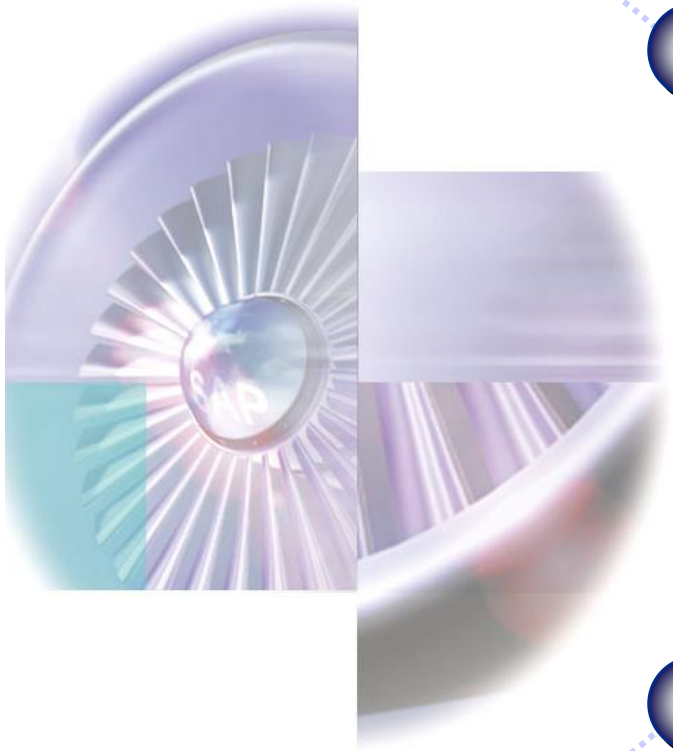
```
        else if(Scores[i] > Second)
        {
            Third = Second;
            Second = Scores[i];
        }
        else if(Scores[i] > Third)
            Third = Scores[i];
        i++;
    }
    printf("前三名成绩分别为：%d，%d，%d\n",
            First, Second, Third);
    return 0;
}
```

运行结果：

```
请输入第 1 个学生的成绩：60
请输入第 2 个学生的成绩：70
请输入第 3 个学生的成绩：80
请输入第 4 个学生的成绩：90
请输入第 5 个学生的成绩：95
请输入第 6 个学生的成绩：100
请输入第 7 个学生的成绩：-1
前三名成绩分别为：100，95，90
```

第1次输入60　Third=0　Second=0　First=60
第2次输入70　Third=0　Second=60　First=70
第3次输入80　Third=60　Second=70　First=80
第4次输入90　Third=70　Second=80　First=90
第5次输入95　Third=80　Second=90　First=95
第6次输入100　Third=90　Second=95　First=100
第7次输入-1　结束
前三名成绩分别为：100，95，90

Lecture 5: 数组



1. 一维数组的定义和引用
2. 二维数组的定义和引用
3. 字符数组
4. 程序举例

Why 二维数组？

- 一维数组：一字长蛇阵
- 队列、巴士、棋盘…

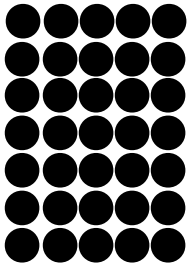
回忆：什么是数组？

定义：一组有序、有名、具有相同数据类型的变量。

- 有名：数组有一个名字，如**RollerCoaster**；
- 有序：数组元素连续存放，**0, 1, 2, …**；
- 同类型：长度相同，如**int, double, char**；
- 变量：数组元素的值可变。

一维数组

二维数组



二维数组的定义方式

类型说明符　　数组名[常量表达式][常量表达式];

例如：int scores[70][16];

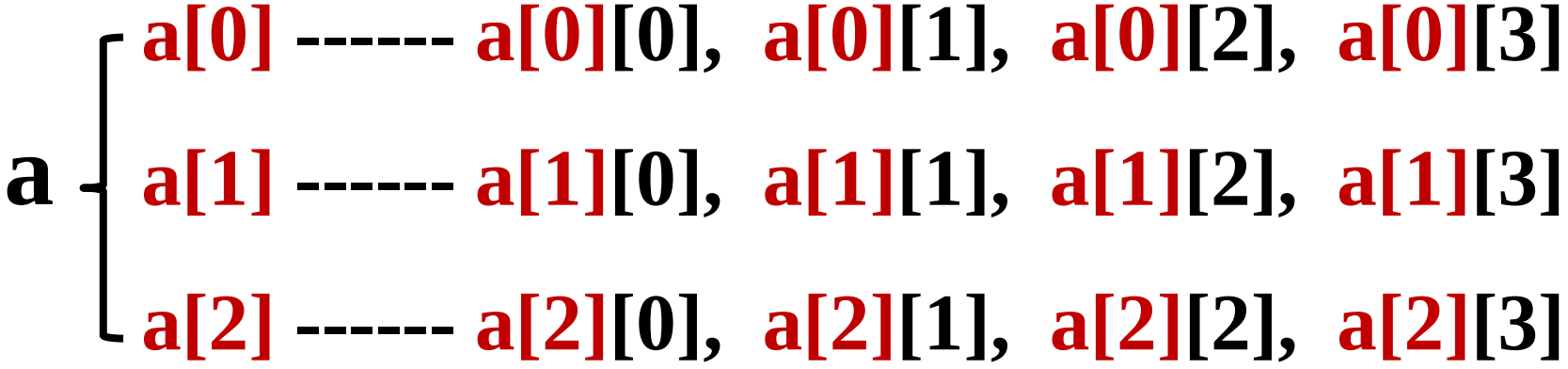
二维数组的一些术语：

int a[3][4];

- **a是一个int类型的二维数组，大小为3×4；**
- **a[0][0]，a[0][1]，a[0][2]，…，a[2][3]都是数组a的元素，每个元素都是一个int类型的变量；**
- **该数组第 i 行、第 j 列的元素为a[i][j]，其中 i 称为数组的行下标，j 称为数组的列下标。**

另一种观点：

二维数组可以看作是一种特殊的一维数组：即它的元素又是一个一维数组。例如，对于int a[3][4]，它有三个元素：a[0]、a[1]和a[2]，每个元素又是一个包含4个元素的一维数组。



```
#include <stdio.h>
int test(int b[ ], int n)
{
    int i;
    for (i=0; i<n; i++)
        printf("%d\n", b[i]);
    return 0;
}

int main()
{
    int a[3][4] = { {0, 1, 2, 3}, {4, 5, 6, 7}, {8, 9, 10, 11} };
    test(a[1], 4);
    return 0;
}
```

```
4
5
6
7
```

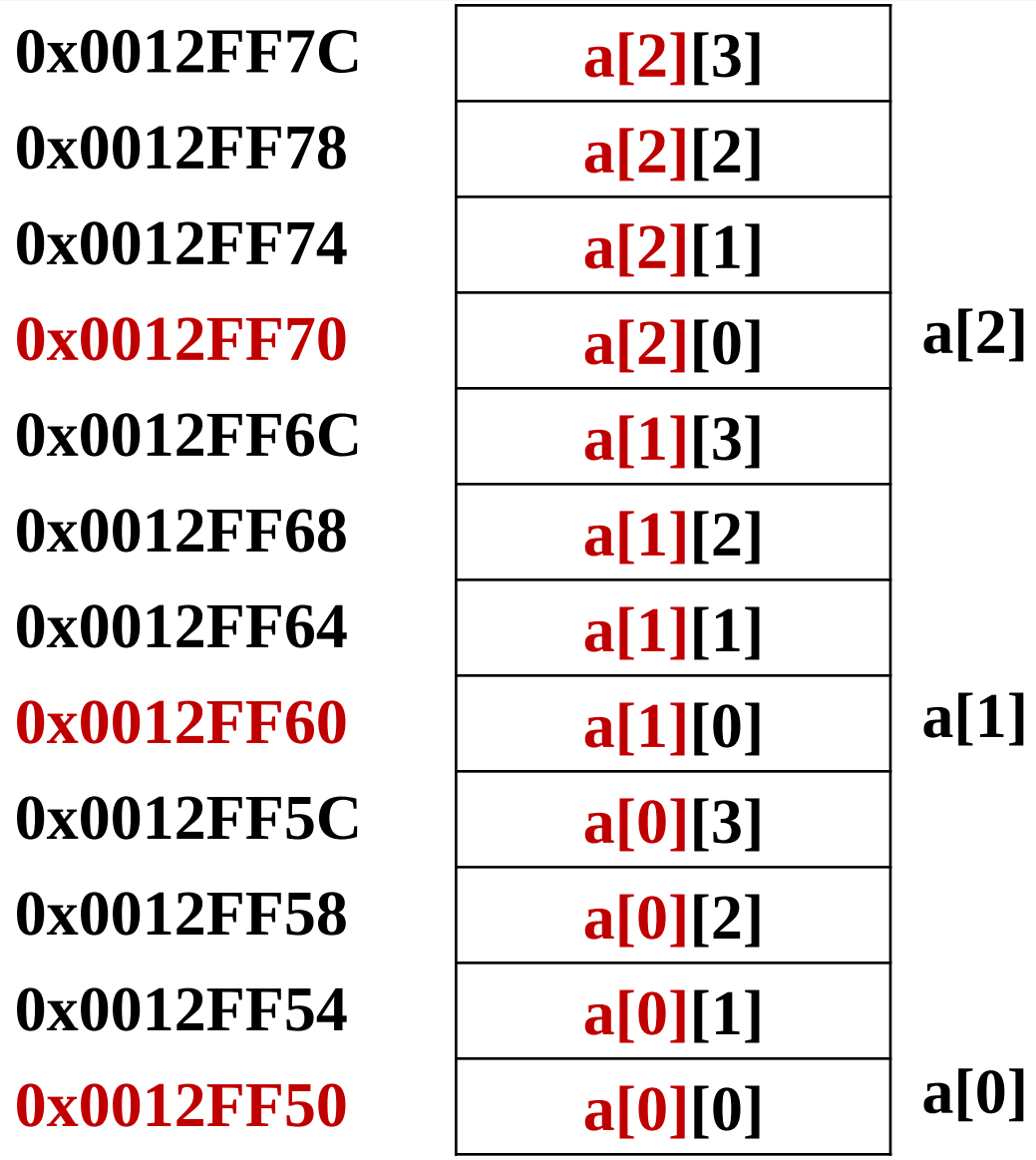
二维数组的内存映像：

二维数组在概念上是二维的。但是，实际的硬件存储器却是一维连续编址的。

如何在一维存储器中存放二维数组，可有两种方式。一种是按行排列，即放完一行之后顺次放入第二行；另一种是按列排列，即放完一列之后再顺次放入第二列。

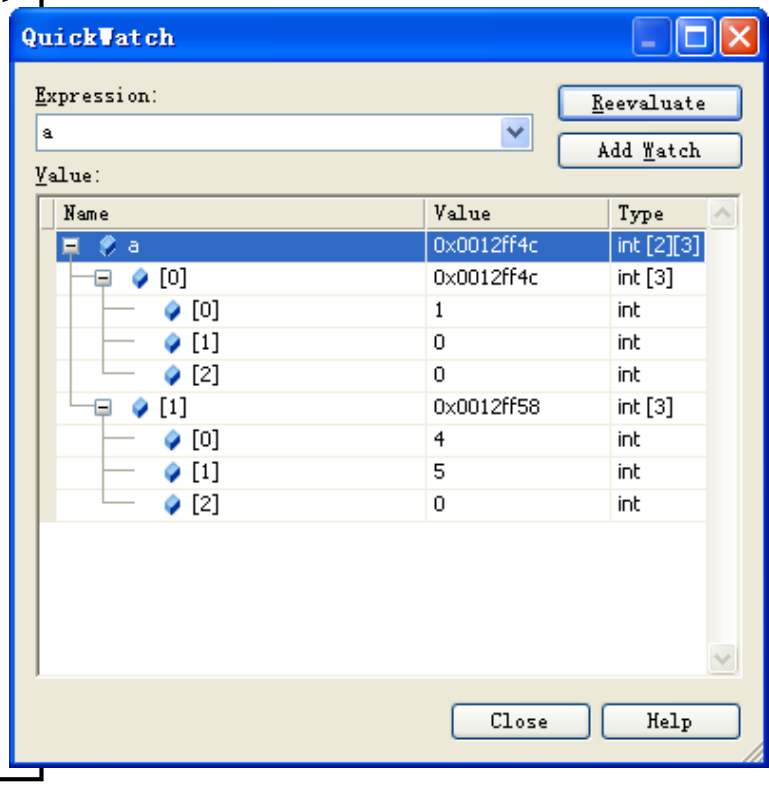
在C语言中，二维数组是按行排列的。

内存分布：
按行存放，
先顺序存放第一行的所有元素，再存放第二行的元素。



二维数组的初始化

- int a[2][3] = { {1, 2, 3}, {4, 5, 6} };
  //按行分段赋值
- int a[2][3] = { 1, 2, 3, 4, 5, 6 };
  //按行连续赋值
- int a[2][3] = { {1}, {4, 5} };
  //部分元素赋值，其他自动赋0值，等价于
  int a[2][3] = {{1, 0, 0}, {4, 5, 0}};



Q1: sizeof(a)?

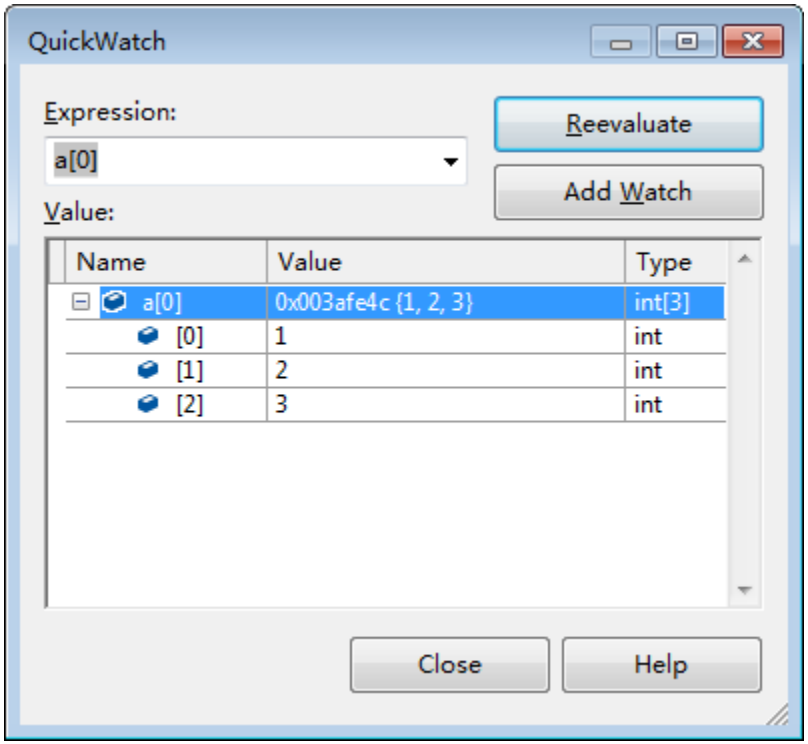
24 = 4*6

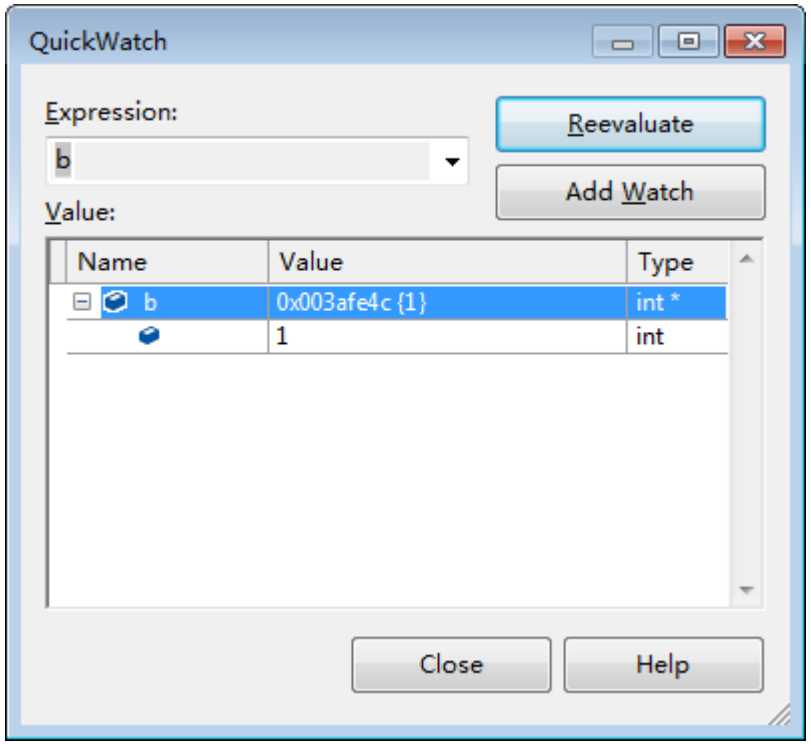
二维数组的初始化

- int a[2][3] = { {1, 2, 3}, {4, 5, 6} };

Q2: a[0] 是什么类型?

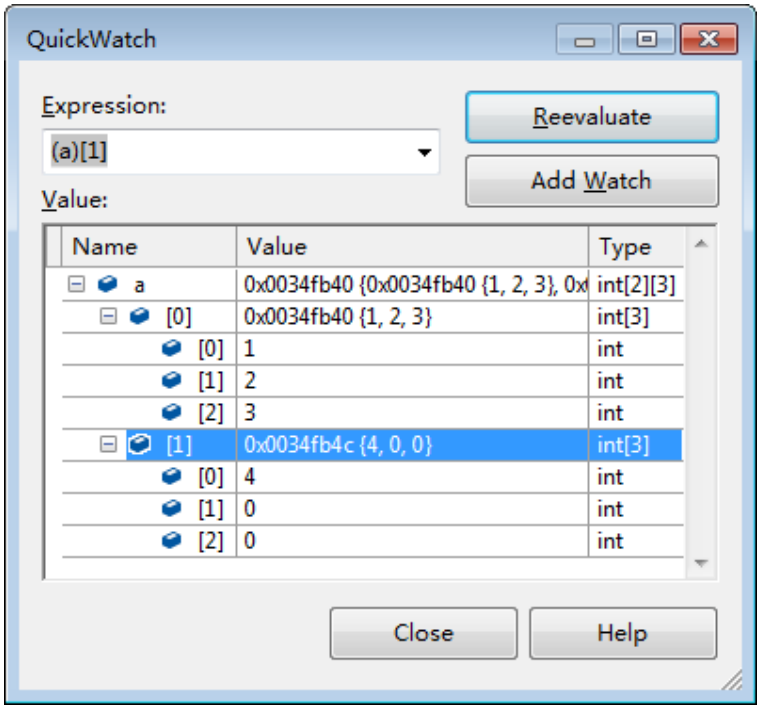
int *b = a[0];





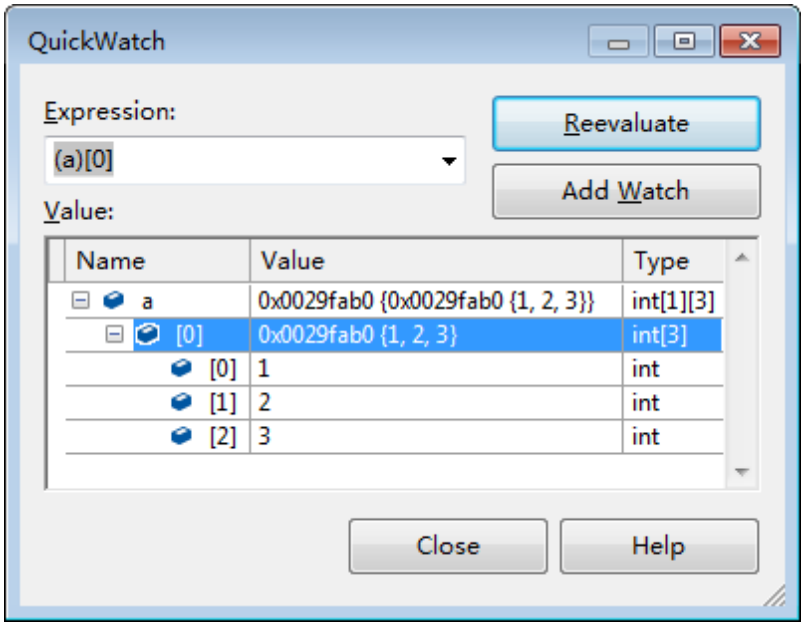
二维数组的初始化

- **int a[ ][3] = { 1, 2, 3, 4, 5, 6 };**
  // 如全部赋值，行可省略
- **int a[ ][3] = { 1, 2, 3, 4 };**
  // 等价于 int a[2][3] = {1, 2, 3, 4, 0, 0};



二维数组的初始化

如果 int a[ ][3] = {1, 2, 3};呢？



不能写成： int a[ ][ ] = { 1, 2, 3, 4, 5, 6 };

或 int a[2][ ] = { 1, 2, 3, 4, 5, 6 };

分析程序

```
int main()
{
    int a[2][3] = {{1,2,3},{4,5,6}};
    int b[3][2], i, j;

    for(i = 0; i <= 1; i++)
      for(j = 0; j <= 2; j++)
          b[j][i] = a[i][j];

    for(i = 0; i <= 2; i++)
    {
        for(j = 0; j <= 1; j++)
            printf("%5d", b[i][j]);
        printf("\n");
    }
    return 0;
}
```

```
1   4
2   5
3   6
```

线性代数中的矩阵转置 $B = A^{T}$

矩阵分解、特征值、特征向量?

补充：变长数组（C99）

- C99标准中新增了变长数组(Variable length arrays)
  - 数组 (a[N]) 的长度可以是一个变量 (int N)
  - 但数组长度变量 (N) 的值必须在runtime 赋值
- C99标准更新：
  - http://www.comeaucomputing.com/techtalk/c99/#bool

- 但VC中的C编译器现在还不支持C99
- 如要在ANSI C中实现可变长度数组，可用动态数组 (malloc( ) / free( ) ， C++是new / delete)，后续课程会介绍

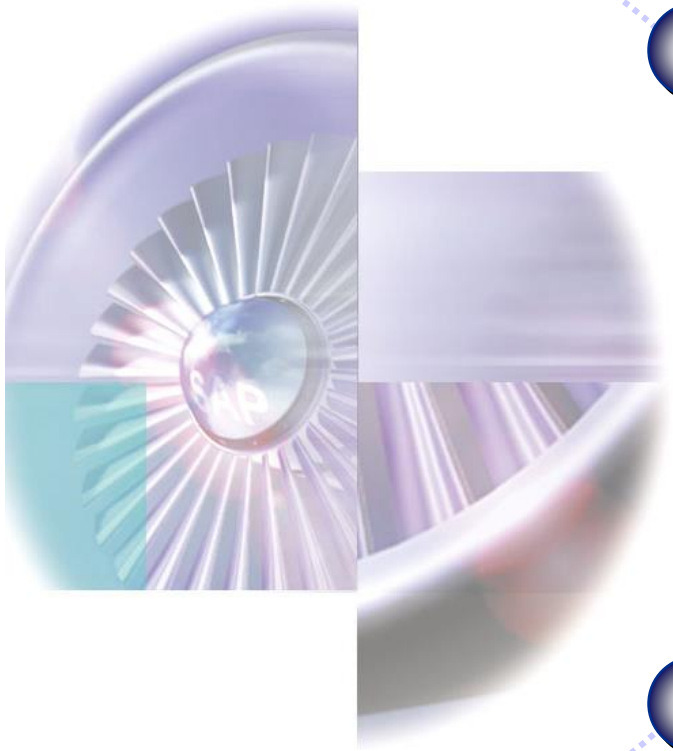
变长数组示例（C99）

```
#include <stdio.h>
int main()
{
    int n, i;
    scanf("%d", &n);
    int a[n];

    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for(i = 0; i < n; i++)
        printf("%5d", a[i]);
    return 0;
}
```

动态数组实现

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n, i;
    scanf("%d", &n);
    int *a = (int *)malloc( n * sizeof(int));

    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for(i = 0; i < n; i++)
        printf("%5d", a[i]);
    free(a);
    return 0;
}
```

Lecture 5: 数组



1. 一维数组的定义和引用
2. 二维数组的定义和引用
3. 字符数组
4. 程序举例

字符数组：用来存放字符数据的数组。

例如：

```
char   c[10] = {'I', ' ', 'a', 'm', ' ', 'h', 'a', 'p', 'p', 'y'};
```

字符数组的定义、初始化和引用与通常的数组都是一样的。

字符串和字符串结束标志：

在 C 语言当中，将字符串作为字符数组来处理，并以字符'\0'来表示字符串的结束标志。

如："How do you do. "

出现“烫烫烫烫”？

H	o	w	␣	d	o	␣	y	o	u	␣	do	.	\0

1. C 语言中字符串为什么以'\0'结尾?
2. 若有多个'\0'呢?
3. C++, Java中的字符串呢?

(int)(0) = = 0

(int)('\0') = = 0

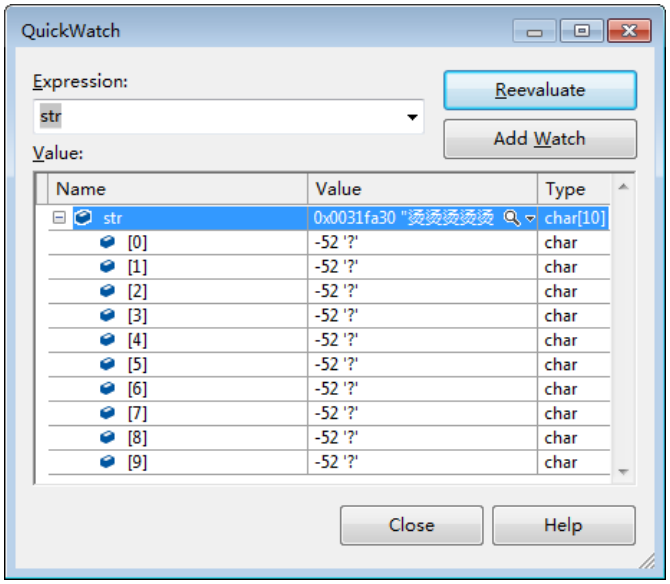
(int)('0') = = 48

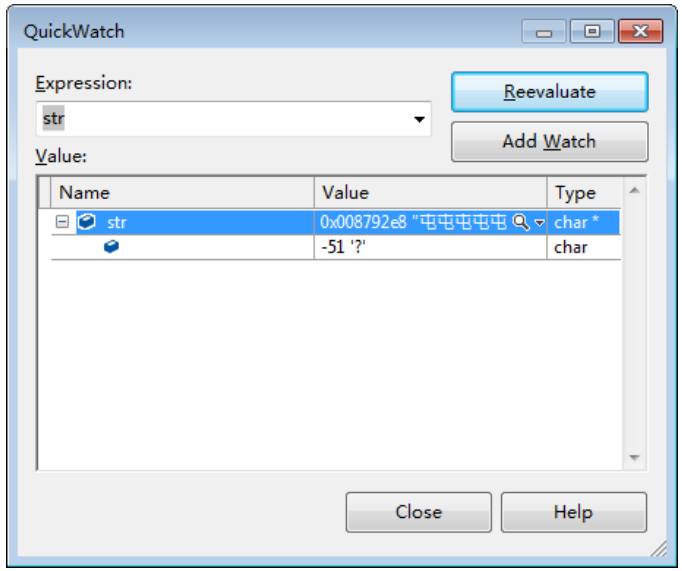
为什么出现了“烫烫烫烫”？

在 Debug 模式下，VC 会把未初始化的栈内存全部填成 0xCC，当字符串看就是“烫烫烫烫……”；会把未初始化的堆内存全部填成 0xCD，当字符串看就是“屯屯屯屯……”。“屯”和“烫”都是GBK编码的结果

```
int main()
{
	char str[10];
	return 0;
}
```

```
int main()
{
	char *str = (char*)malloc(10*sizeof(char));
	return 0;
}
```





字符数组初始化

```
char str[ ] = {"Hello"};
```

整体赋值只能在字符数组初始化时使用

可以省略花括号：

```
char str[ ] = "Hello";
```

等价于

```
char str[ ] = {‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’};
```

用字符指针指向一个字符串

```
char *str = "Hello";
```

```
#include <stdio.h>
int main()
{
    char *str;
    str = "Hello";
    printf("%s\n", str);
    return 0;
}
```

√

```
#include <stdio.h>
int main()
{
    char str[6];
    str = "Hello";
    printf("%s\n", str);
    return 0;
}
```

×

为什么不可以？

error C2440: '=' : cannot convert from 'const char [6]' to 'char [6]'

字符串处理函数

（一）puts函数（字符串输出）

功能：将一个字符串输出到屏幕。

格式：**int puts(const char *string);**

例子：
```
char str[ ] = {"Hello"};
puts(str);
puts("Hello");
```

（二）gets函数（字符串输入）

功能：从键盘输入一个字符串到字符数组。

格式：**char *gets(char *buffer);**

例子：

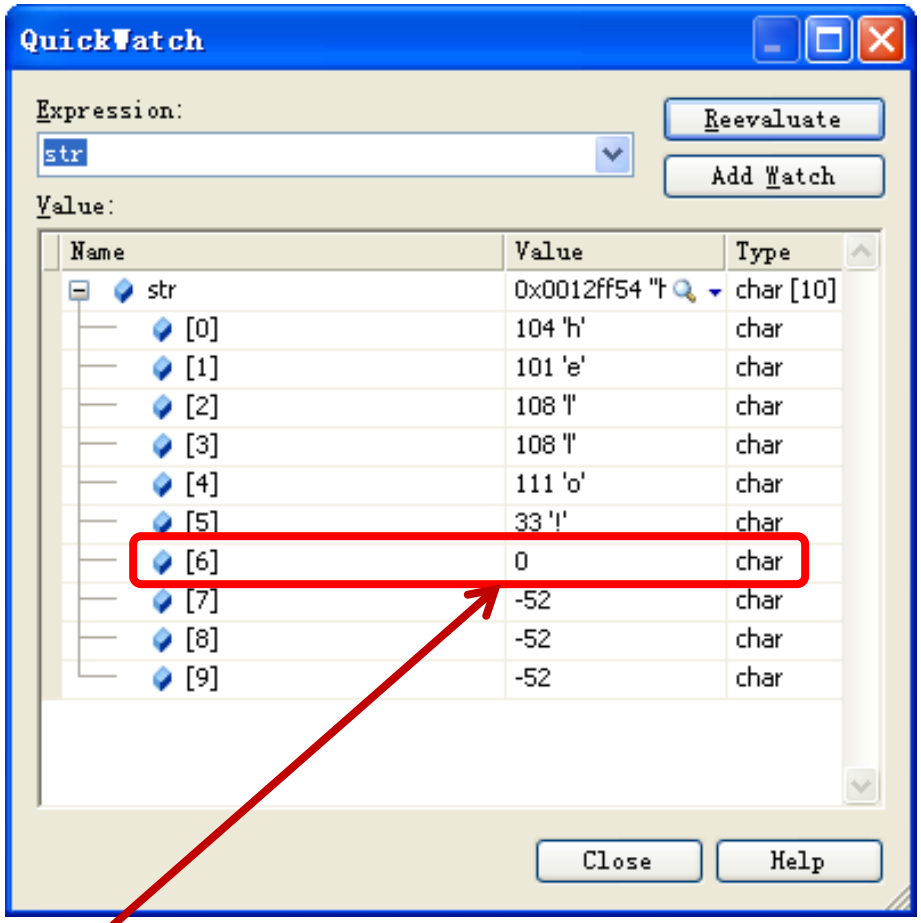
```
char str[10];
gets(str);
```

自动补零！

gets()与scanf() 的区别？

```
#include <stdio.h>
int main()
{
    char  str[10];
    gets(str); // input "hello!"
    puts(str);
    return 0;
}
```

```
hello!
hello!
```



自动补零！

gets()和scanf()函数的区别

gets()函数读取字符串，直到碰上第一个**换行符**为止；

scanf()函数读取字符串，遇到**空格、回车和Tab键**都会认为输入结束。

```
#include <stdio.h>
int main()
{
    char  str[20];
    gets(str); // input "hello world!"
    puts(str);
    scanf("%s", str); // input "hello world!"
    puts(str);
    return 0;
}
```

```
hello world!
hello world!
hello world!
hello
```

如果gets和scanf互换会怎么样？

fflush(stdin) 功能: 清空输入缓冲区，通常是为了确保不影响后面的数据读取。如在读完一个字符串后紧接着又要读取一个字符，此时应该先执行**fflush(stdin)**。

```
#include <stdio.h>
int main()
{
    char  str[20];
    scanf("%s", str); // 输入"hello world!"
    puts(str); // 输出"hello"
    //fflush(stdin);
    gets(str); // 直接从缓冲区获得" world!"
    puts(str); // 输出" world!"
    return 0;
}
```

```
hello world!
hello
 world!
```

字符串的输入方法总结

- **scanf("%s"，str)**：输入一个字符串，直到碰上第一个**空白字符(空格、Tab或换行)**；
- **gets(str)**：输入一行字符，直到碰上第一个**换行**；
- **scanf输入的字符串不能包含空格和Tab，而gets输入的字符串可以。若分别使用这两种不同的输入方法，则当用户敲入“aa bb cc”时，得到的输入字符串分别是：“aa”和“aa bb cc”；**
- **fflush(stdin)**：清空输入缓冲区 (**stdin**)。
  - **Standard streams (标准流) for input (stdin), output (stdout), and error output (stderr).**

scanf 函数补充说明

- 功能：从键盘输入若干个各种类型的数据。
- 格式：scanf(“格式控制字符串”， 地址列表);
  - “格式字符说明”：%s 读入一个字符串，遇空格、制表符或换行符结束。
  - %[ ] 扫描字符集合。scanf中一种很有用的转换字符：[...] 和 [^...]。
- 示例：
  - scanf("%[^\n]", str); // 回车结束输入(可以读取空格)
  - scanf("%[1234567890]", str);

```c
#include <stdio.h>
int main()
{
    char str[80];
    scanf("%[^\n]", str);
    printf("%s", str);
    return 0;
}
```

```
hello world!
hello world!
```

```c
#include<stdio.h>
int main()
{
    char str[80];
    scanf("%[1234567890]", str);
    printf("%s", str);
    return 0;
}
```

```
1234test5678
1234
```

（三）strcat函数（字符串连接）

cat: catenate, catenation

格式：char *strcat(char *szDestination,
const char *szSource);

功能：连接两个字符数组中的字符串，把szSource连接到szDestination的后面。

例子：
```
char szSrc[ ] = "world";
char  szDest[30] = "hello ";
strcat(szDest,  szSrc);   // "hello  world"
```

头文件：#include <string.h>

（四）strcpy函数（字符串复制）

格式：**char *strcpy(char *szDestination,**
const char *szSource);

功能：将字符串**szSource**复制到字符数组**szDestination**中去。

例子：
```
char  szDest[30];
char szSrc[ ] = "hello";
strcpy(szDest,  szSrc);
```

```
#include <string.h>
#include <stdio.h>
int main()
{
        char string[80];
        strcpy( string, "Hello world from " );
        strcat( string, "strcpy " );
        strcat( string, "and " );
        strcat( string, "strcat!" );
        printf( "String = %s\n", string );
        return 0;
}
```

String = Hello world from strcpy and strcat!

（五）strcmp函数（字符串比较）

格式：int strcmp(const char *string1,
const char *string2);

功能：比较字符串string1和string2。

说明：如果string1 < string2, 返回一个负整数(-1);
如果string1 = string2, 返回0;
如果string1 > string2, 返回一个正整数(1);

例子：strcmp(str1, str2);
strcmp("America", "China");
strcmp("Love", "Life");
strcmp("ab", "abc");

- **strcmp函数是比较字符串中各对字符的ASCII码。**
- **首先比较两个串的第一个字符，若不相等，则停止比较并得出大于或小于的结果；**
- **如果相等，就接着比较第二个字符，然后第三个字符等等；**
- **strcmp函数最多比较到其中一个字符串遇到结束符‘/0’为止；**
- **strcmp("disk"，"disks"); // -1**

```
#include <string.h>
#include <stdio.h>
int main()
{
        char *s1 = "aB";
        char *s2 = "Ab";
        char *s3 = "aa";
        char *s4 = "aB";
        printf("%s vs %s: %d\n", s1, s2, strcmp(s1, s2));
        printf("%s vs %s: %d\n", s1, s3, strcmp(s1, s3));
        printf("%s vs %s: %d\n", s1, s4, strcmp(s1, s4));
        return 0;
}
```

```
aB vs Ab: 1

aB vs aa: -1

aB vs aB: 0
```

strlen函数： 返回字符串的实际长度（不包括末尾的‘\0’字符）。

strlwr函数： 把字符串中的大写字母换成小写字母（lowercase）。

strupr函数： 把字符串中的小写字母换成大写字母（uppercase）。

……

选择题

下列语句中，不能把字符串“Hello!”赋给字符数组b的语句是（C）

A. char b[10] = {'H', 'e', 'l', 'l', 'o', '!'};

B. char b[10]; strcpy(b, "Hello!");

C. char b[10]; b = "Hello!";

D. char b[10] = "Hello!";

填空

```
/* 程序功能是将无符号八进制数字构成的字符串转换为十进制整数。如:若输入字符串1234，则输出十进制整数668 */
int main( )
{
    char s[10];
    int i, n = 0;
    gets(s);
    i = 0;
    while( s[i] != '\0' )
    {
        n = n*8 + (s[i] - '0‘);
        i++;
    }
    printf("%d\n", n);
    return 0;
}
```

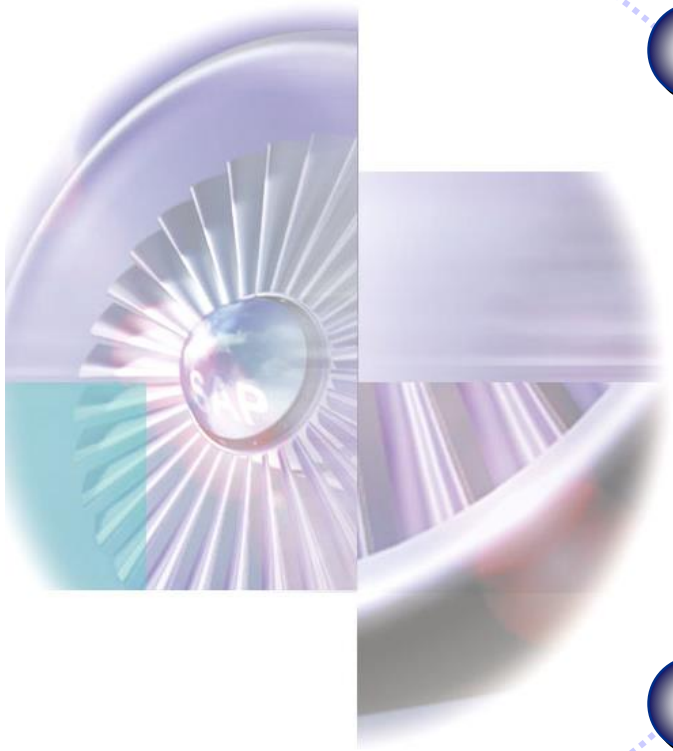
分析运行结果

```
int main( )
{
    char word[] = "ijklmnop";
    char s[100];
    int len, i, j = 0;
    char c1, c2;
    len = strlen(word);
    for (i = 0; i < len / 2; i++)
    {
        if (i < 2) { c1 = word[3];  c2 = word[6-3*i];}
        else  { c1 = word[6*(i-2)];  c2 = word[7]; }
        s[j++] = c1;
        s[j++] = c2;
    }
    s[j] = '\0';
    puts(s);
    return 0;
}
```

lollipop

棒棒糖！
Android 5.0 Lollipop

Lecture 5: 数组



1. 一维数组的定义和引用
2. 二维数组的定义和引用
3. 字符数组
4. 程序举例

程序举例

- **示例1:字符串拼接**
- **示例2:回文问题**
- **示例3:冒泡排序**
- **示例4:找数**
- **示例5:DNA序列**

Lecture 5 - Summary

- **Topics covered:**
  - **The concept of array**
  - **Defining arrays**
  - **Initializing arrays**
  - **Character arrays**
  - **Multidimensional arrays**
  - **Variable length arrays**