> 必备的习惯： 感恩 自律 阅读 锻炼
负责的习惯： 不责备 不抱怨 不多解释
快乐的习惯： 不生气 不嫉妒 不多想 不说人非

> 人生在关键的时刻一定要做对的事！！
> 复盘 方法 高效

@[toc]

第07次直播 选择与循环

关系表达式与逻辑表达式

- 各个运算符的优先级顺序如下

1. 单目运算符 `！` 逻辑非
2. 算术运算符　加、减，乘除，取模
3. 关系运算符 `> <　>=　<`
4. 逻辑运算符 `&&　||`

例子：

在介绍选择语句前，我们首先练习一下关系表达式与逻辑表达式。在第 2 章中，我们了解到算术运算符的优先级高于关系运算符、关系运算符的优先级高于逻辑与和逻辑或运算符、相同优先级的运算符从左至右进行结合等，那么表达式 5>3&&8<4-!0 的最终值是多少？其计算过程如图 3.1.1 所示。

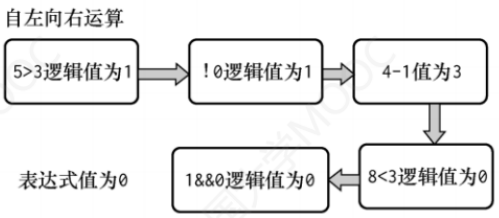


图 3.1.1　5>3&&8<4-!0 的计算过程

关于 if 语句使用

注意一： If 语句的写法。注意事项

```
int main()
{
    int i;
    while (scanf("%d", &i) != EOF)
    {
        if (i > 0);
        {
            printf("i is bigger than 0\n");
        }
    }
}
```


- 不能按上述写法编写，否则下面的 `I is bigger than 0` ,无论输入什么条件都会一直执行。。

关于if 语句后面加大括号问题

```
int main()
{
    int i;
    while (scanf("%d", &i) != EOF)
    {
        if (i > 0)//在if的括号后面不可以加;，会造成表达式无论是真还是假，
            printf("i is bigger than 0\n");
            printf("i is bigger than 0
            printf("i is bigger than 0
    }
}
```

```
D:\Book\第三章 选择与循环\Debug\1-if与else.exe
20
i is bigger than 0
i is bigger than 0
i is bigger than 0
-5
i is bigger than 0
i is bigger than 0
```


while 语句

- 当命令行（黑窗口），只有两种情况，①一种是卡在 `scanf（` 这种情况下手输入内容，是可以输入进去的），②另外一种是死循环
- 在 Windows操作系统下的VS集成开发环境中，我们可以用 `ffush` 或 `rewind` 清空标准输入缓冲区，但是这些函数在 Linux操作系统中是无法使用的。自己实现一个清空缓冲区的

```
while((ch=getchar())!=EOF && ch!='\n') ;
```

for 循环的使用

for(表达式 1;表达式 2;表达式 3) 语句;

for 循环语句的执行过程如下，具体流程如图 3.2.3 所示。

（1）先求解表达式 1。

（2）求解表达式 2，若其值为真（值为非 0），则先执行 for 语句中指定的内嵌语句，后执行第（3）步。若其值为假（值为 0），则结束循环，转到第（5）步。

（3）求解表达式 3。

（4）转回第（2）步继续执行。

（5）循环结束，执行 for 语句下面的语句。

while 语句中出现死循环的原因

- 1、 `while()` 后面加了分号
- 2、 while语句体内没有让表达式趋近于假的操作

注意事项： 关于在for() 语句后面加上 `;`

```
//从1加到100
int main()
{
    int i, total;
    //for语句中只能有两个分号
    for (i = 1, total = 0; i <= 100; i++);
    {
        total = total + i;
    }
    printf("total=%d\n", total);
}
```


预期的结果为： 101

continue 、break语句

- continue **跳出本轮循环**，即为从代码后面的语句不会执行

```
int main()
{
    int i, total;
    //for语句中只能有两个分号
    for (i = 1, total = 0; i <= 100; i++)//for循环后不能加分号
    {
        if (i % 2==0)//如何i是偶数
        {
            continue;//提前结束本轮循环
        }
        total = total + i;
    }
    printf("total=%d\n", total);
```

跳过不会运行

- `break` 是结束整个循环

```
int main()
{
    int i,total;
    for(i=1,total=0;i<=100;i++)
    {
        if(total>2000)
        {
            break;//for循环结束，下一条语句是printf
        }
        total=total+i;
    }
    printf("total=%d,i=%d\n",total,i);
    system("pause");
```


第08次直播 数组 字符串数组

一维数组的表示

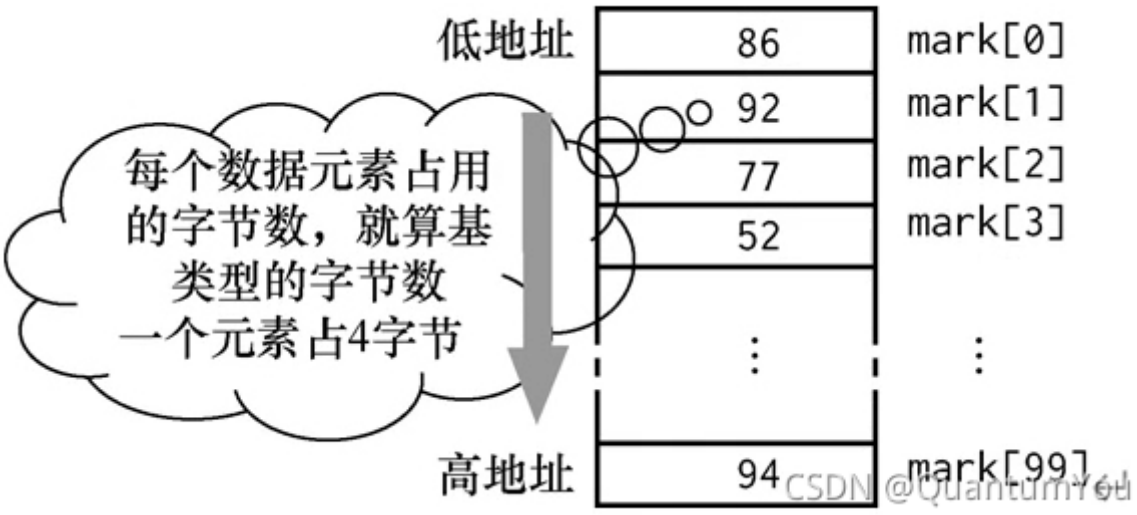
- 在C语言中一维数组的定义格式为：

```
类型说明符 数组名 [常量表达式3] ;   int a[10];
```

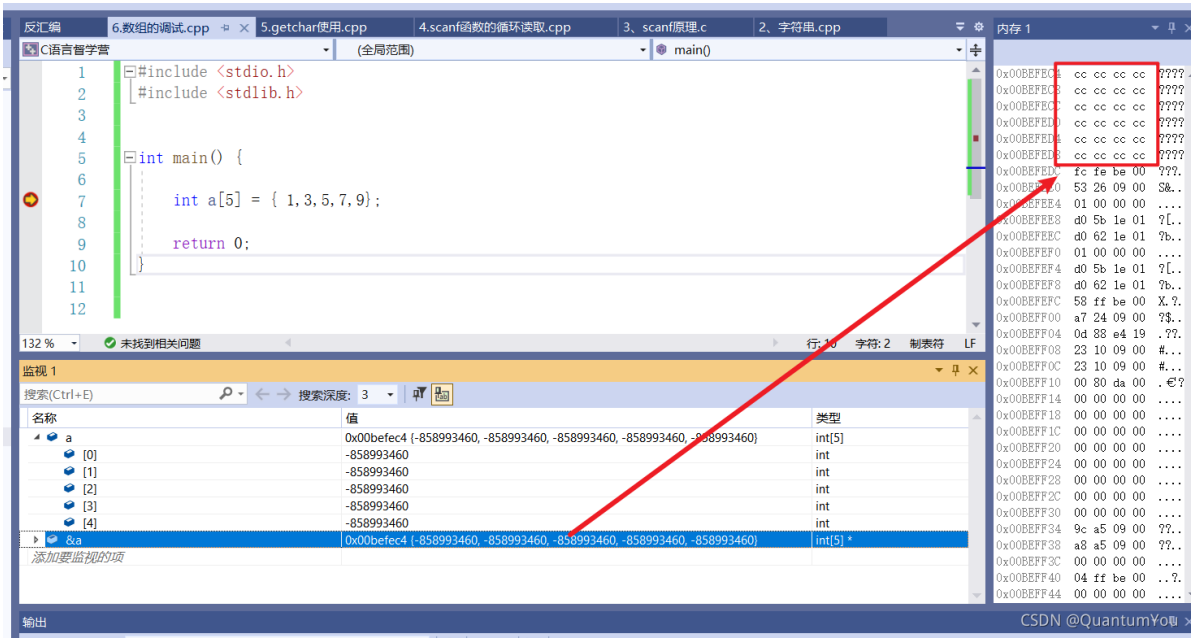
声明数组时要遵循以下规则：

- (1) 数组名的命名规则和变量名的相同，即遵循标识符命名规则
- (2) 在定义数组时，需要指定数组中元素的个数，方括号中的常量表达式用来表示元素的个数，即数组长度。
- (3) 常量表达式中可以包含常量和符号常量，但不能包含变量。也就是说，**C语言不允许对数组的大小做动态定义**，即数组的大小不依赖于程序运行过程中变量的值。

一维数组在内存中的存储



进行调试：

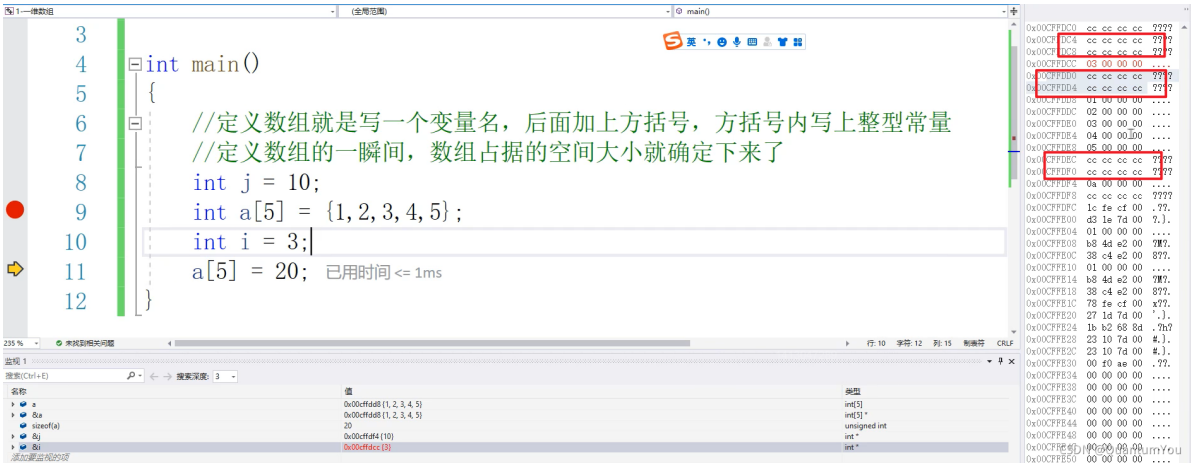


小细节解析

存储空间分析

- 在微软中，先定义的变量存储于高地址
- 看任何变量的内存，都是将该变量取地址，拖入内存窗口来查看
- 微软的编译器设计，不同的变量之间有8个字节的保护空间（不是C标准）

微软编译器设计有8个字节的变量保护空间



数组打印易错分析

```c
//打印数组里的每一个元素
void print(int a[])
{
    int i;
    for (i = 0; i < sizeof(a)/sizeof(int); i++)
    {
        printf("a[%d]=%d\n", i, a[i]);
    }
}
```


- 上述的 `sizeof (a) / sizeof(int)` 仍然为1的原因为 `print()` 传递的数组个数为无法传递，只能传递首地址（数组的起始地址）

正确写法如下

```c
//打印数组里的每一个元素,数组在传递时，元素个数传递不过去
void print(int a[],int len)
{
    int i;
    for (i = 0; i < len; i++)
    {
        printf("a[%d]=%d\n", i, a[i]);
    }
}
```


> 注意： 在子函数中修改数组，原数组中也相应改变

```c
#include <stdio.h>

//打印数组里的每一个元素,数组在传递时，元素个数传递不过去
void print(int b[],int len)
{
    int i;
    for (i = 0; i < len; i++)
    {
        printf("a[%d]=%d\n", i, b[i]);
    }
    b[4] = 20;//在子函数中修改数组元素
}
```


字符数组

字符数组进行赋值

- 一般一个一个赋值较为麻烦，简写方式如下：

```
char c[10] = {'h','e','l','l','o'};
```

```
char c[10] = "hello";
```


关于字符串数组常见错误

- 字符串输出乱码问题

```
#include <stdio.h>

int main()
{
    char c[5] = { 'h','e','l','l','o' };
    char d[5] = "how";
    printf("%s----%s\n", c, d);//printf的%s，对应后面要写的是字符数组名，字
}
```

```
选择Microsoft Visual Studio 调试控制台
hello烫烫烫?Y?烫?---how

D:\Book\第四章 数组\Debug\2-字符数组.exe (进程 30708)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
```


- hello 后面打出乱码“烫烫烫” 的原因，没有想 how 一样遇到结束符。

```
#include <stdio.h>

int main()
{
    char c[5] = { 'h','e','l','l','o' };
    char d[5] = "how";
    printf("%s----%s\n", c, d);//printf的%s，对应后面要写的是字符数组名，
}
```

结束符

```
0x004FF9C8  cc cc cc cc  ????
0x004FF9CC  cc cc cc cc  ????
0x004FF9D0  cc cc cc cc  ????
0x004FF9D4  68 6f 77 00  how.
0x004FF9D8  00 cc cc cc  .???
0x004FF9DC  cc cc cc cc  ????
0x004FF9E0  cc cc cc cc  ????
0x004FF9E4  68 65 6c 6c  hell
0x004FF9E8  6f cc cc cc  o???
0x004FF9EC  cc cc cc cc  ????
0x004FF9F0  04 10 a6 64  ..?d
0x004FF9F4  14 fa 4f 00  .?O.
0x004FF9F8  03 1f 54 00  ..T.
0x004FF9FC  01 00 00 00  ....
0x004FFA00  58 4e 8e 00  XN?.
0x004FFA04  b8 c5 8e 00  ???.
0x004FFA08  01 00 00 00  ....
0x004FFA0C  58 4e 8e 00  XN?.
0x004FFA10  b8 c5 8e 00  ???.
0x004FFA14  70 fa 4f 00  p?O.
0x004FFA18  57 1d 54 00  W.T.
0x004FFA1C  80 13 a6 64  €.?d
0x004FFA20  23 10 54 00  #.T.
0x004FFA24  23 10 54 00  #.T.
0x004FFA28  00 30 2f 00  .0/.
0x004FFA2C  00 00 00 00  ....
```


- **总结结论**：在字符串初始化时一定要多初始化字节（用于存储结束符），`hello` 正好五个没有预留，how 三个足以存储。
- 因为C语言规定字符串的结束标志为 `'\0'`
- 上述正确写法为 `char c[6] = {'h','e','l','l','o'} ;`

字符串的输入

```
//初始化字符数组时，一定要让字符数组的大小比看到的字符串的长度多1
int main()
{
    char c[6] = { 'h','e','l','l','o' };
    char d[5] = "how";
    printf("%s----%s\n", c, d);//printf的%s，对应后面要写的是字符数组名，
    char e[20];
    scanf("%s", e);
    printf("%s\n", e);
}
```

已用时间 <= 7,975ms

这里的字符接受 不需要取地址符 &

```
0x010FF9C4  77 61 6e 67  wang
0x010FF9C8  64 61 6f 00  dao.
0x010FF9CC  cc cc cc cc  ????
0x010FF9D0  cc cc cc cc  ????
0x010FF9D4  cc cc cc cc  ????
0x010FF9D8  cc cc cc cc  ????
0x010FF9DC  cc cc cc cc  ????
0x010FF9E0  68 6f 77 00  how.
0x010FF9E4  00 cc cc cc  .???
0x010FF9E8  cc cc cc cc  ????
0x010FF9EC  cc cc cc cc  ????
0x010FF9F0  68 65 6c 6c  hell
0x010FF9F4  6f 00 cc cc  o.??
0x010FF9F8  cc cc cc cc  ????
0x010FF9FC  75 4f 13 73  uO.s
0x010FFA00  20 fa 0f 01   ?..
0x010FFA04  23 21 ba 00  #!?.
0x010FFA08  01 00 00 00  ....
0x010FFA0C  58 4e 3e 01  XN?.
0x010FFA10  e0 71 3e 01  ?q?.
0x010FFA14  01 00 00 00  ....
0x010FFA18  58 4e 3e 01  XN?.
0x010FFA1C  e0 71 3e 01  ?q?.
0x010FFA20  7c fa 0f 01  |?..
0x010FFA24  77 1f ba 00  w.?.
0x010FFA28  09 4f 13 73  .O.s
0x010FFA2C  23 10 ba 00  #.?.
0x010FFA30  23 10 ba 00  #.?.
0x010FFA34  00 b0 e2 00  .??.
0x010FFA38  00 00 00 00  ....
0x010FFA3C  00 00 00 00  ....
0x010FFA40  00 00 00 00  ....
0x010FFA44  00 00 00 00  ....
0x010FFA48  00 00 00 00  ....
0x010FFA4C  00 00 00 00  ....
0x010FFA50  00 00 00 00  ....
```


- 注意上述输入会自动添加结束符。