# 第一单元:

## 单元练习目的

学生在掌握了计算机对于程序的编译和执行的基本原理后,使用以微软公司的IDE程序开发平台Visual Studio完成练习。以此使学生能够比较深刻的了解程序执行的基本方式和原理，可以熟练的使用各种调试工具进行程序的调试，领悟函数，变量和指针的本质，并且对于程序对于内存的使用有一个初步的认识。

### 练习1

利用VisualStudio的工程建立向导，建立一个Helloworld程序。利用鼠标右键功能菜单或者F9按键，加入断点进行调试。将main函数的两个参数argc和argv利用鼠标右键添加到监视窗口，观察argc和argv的内容，了解argc的数值和argv的内容有什么关系？在工程-属性里的调试页面内，添加数值进入命令参数，再次观察argc和argv的内容有何变化。（提示：argv默认是指向argv[0]的地址，如果要查看argv的后续第n项内容需要输入argv[n]）

### 练习2

建立一个工程，在源文件中加入下列代码：

#include <stdio.h>

char globalchar1;

char globalchar2 = 'g';

int globalint1;

int globalint2 = 9;

char globalchar3;

int main (int argc, char \* argv[])

{

char localchar1;

char localchar2 = 'l';;

int localint1;

int localint2 = 1;

char localchar3;

printf("Globals: '%c'(%d) '%c'(%d) %d %d '%c'(%d)\n",

globalchar1, globalchar1,

globalchar2, globalchar2,

globalint1, globalint2,

globalchar3, globalchar3);

printf("Locals: '%c'(%d) '%c'(%d) %d %d '%c'(%d)\n",

localchar1, localchar1,

localchar2, localchar2,

localint1, localint2,

localchar3, localchar3);

return 0;

}

将globalchar1，globalchar2，globalint1，globalint2，globalchar3以及localchar1, ocalchar2,localint1, localchar3添加到监视窗口，观察这些变量的内存地址有何特征。

(提示:全局变量和局部变量的内存分配的区别)

### 练习3

建立一个工程，在源文件中加入下列代码：

#include <stdio.h>

int main (int argc, char \* argv[])

{

int a,\*p,\*\*q;

p=&a;

q=&p;

int b,c,\*r;

b=c=0;

r=&b;

c=\*r;

}

将a，b，c和p,q,r添加到监视窗口，观察它们代表的数值之间有何种联系。

(提示:注意变量和指针的区别)

### 练习4

建立一个VC的Win32控制台工程，将下列代码加入一个源文件内。

#define ARRAY\_SIZE 10

void natural\_numbers (void) {

int i;

int array[ARRAY\_SIZE];

i = 1;

while (i <= ARRAY\_SIZE) {

array[i] = i - 1;

i = i + 1;

}

}

完成代码编写后进行调试，看看程序在执行的过程中会发生什么情况。（提示：将变量“i”和数组array加入监视窗口内观察结果,尤其是程序在array边界的执行情况）

### 练习5

建立一个工程，在源文件中加入下列代码：

#include <stdio.h>

void Initialize (char \* a, char \* b)

{

a[0] = 'T'; a[1] = 'h'; a[2] = 'i';

a[3] = 's'; a[4] = ' '; a[5] = 'i';

a[6] = 's'; a[7] = ' '; a[8] = 'A';

a[9] = '\0';

b = a;

b[8] = 'B';

}

#define ARRAY\_SIZE 10

char a[ARRAY\_SIZE];

char b[ARRAY\_SIZE];

int main(int argc, char \* argv[])

{

Initialize(a, b);

printf("%s\n%s\n", a, b);

return 0;

}

观察程序运行过程中字符串数组a和b的内容有什么变化，这种变化发生的原因是什么？