# 第三单元:

## 单元练习目的

学生在掌握计算机对于程序的内存的分配和使用方式后,能够了解程序进行动态和静态内存分配的机制和方法,并且在编写程序中能够避免出现内存泄露和不正确回收等问题出现。

### 练习1

将下面的两段代码进行调试,观察程序对于静态变量的定义和初始化方式:

代码段A:

int my\_var[128]; // a statically allocated variable

int my\_fn(int x)

{

// note that the initialization of my\_var\_initialized

// takes place only once, not every time my\_fn() is called:

static bool my\_var\_initialized = false;

if (my\_var\_initialized) return;

my\_var\_initialized = true;

for (int i = 0; i < 128; i++)

my\_var[i] = 0;

}

代码段B:

int my\_var[128]; // a statically allocated variable

static bool my\_var\_initialized = false;

int my\_fn(int x)

{

if (my\_var\_initialized) return;

my\_var\_initialized = true;

for (int i = 0; i < 128; i++)

my\_var[i] = 0;

}

### 练习2

在工程里加入以下代码，在进行调试的过程中,观察堆栈的使用情况,了解程序是如何调用堆栈和返回的。

int foo()

{

int b;

b = bar();

return b;

}

int bar()

{

int b = 0;

b = baz(b);

return b;

}

int baz(int b)

{

if (b < 1) return baz(b + 1);

else return b;

}

### 练习3

建立一个工程，在源文件中加入下列代码，了解在堆上手动分配内存和回收内存的方式和运行机制：

int var;

int \*var\_ptr;

var\_ptr = &var;

\*var\_ptr = 3;

if (var == \*var\_ptr) printf("ok\n");

var\_ptr = (int \*) malloc(sizeof(int));

\*var\_ptr = 4;

free(var\_ptr);

typedef struct {

int int\_field;

double dbl\_field;

} my\_struct\_type;

my\_struct\_type \*s;

s = (my\_struct\_type \*) malloc(sizeof(my\_struct\_type));

s->int\_field = 0;

s->dbl\_field = 0.0;

(\*s).int\_field = 0;

### 练习4

下面是几段对于内存操作错误的代码，请分析错误原因并找出解决办法：

代码段A：

#define array\_size 100

int \*a = (int \*) malloc(sizeof(int \*) \* array\_size);

for (int i = 0; i <= array\_size; i++)

a[i] = NULL;

代码段B：

#define array\_size 100

int \*a = (int \*) malloc(array\_size);

a[99] = 0; // this overwrites memory beyond the block

代码段C：

char \*heapify\_string(char \*s)

{

int len = strlen(s);

char \*new\_s = (char \*) malloc(len);

strcpy(new\_s, s);

return new\_s;

}

代码段D：

char q[] = "do not overflow";

char r[] = " memory";

char s[16];

strcpy(s, q);

strcat(s, r);

代码段E：

strcat("do not overflow", " memory");

代码段F：

void dec\_positive(int \*a)

{

\*a--; // decrement the integer

if (\*a < 0) \*a = 0; // make sure a is positive

}

代码段G：

int \*ptr\_to\_zero()

{

int i = 0;

return &i;

}

### 练习5

下面两段代码都存在对于内存回收的问题，找出问题所在，并提出解决办法：

代码段A：

void my\_function(char \*msg)

{

// allocate space for a string

char \*full\_msg = (char \*) malloc(strlen(msg) + 100);

strcpy(full\_msg, "The following error was encountered: ");

strcat(full\_msg, msg);

if (!display(full\_msg)) return;

free(full\_msg);

}

代码段B：

typedef struct {

char \*name;

int age;

char \*address;

int phone;

} Person;

void my\_function()

{

Person \*p = (Person \*) malloc(sizeof(Person));

p->name = (char \*) malloc(128);

p->address = (char \*) malloc(128);

free(p);

}