# 第0单元:

## 单元练习目的

本单元帮助学生理解、完善C语言的基本概念、原则，通过具体案例，消化吸收课堂所讲的内容，厘清C语言中容易混淆的概念，区分C语言、C++、Java等语言的不同点，为后续的学习、实验打好基础。主要概念点有：

* C语言的基本调试环境
* C语言中关键概念的使用，以及其存在的缺陷；
* 指针与数组的概念与关系
* 数据结构

### 练习1

本课程使用ANSI C为标准。

通过参考书、网络资源等，了解ANSI C的基本术语、概念、定义等。

### 练习2

该练习要求学生根据自己的喜好，准备C语言的编辑、调试环境和工具。

学院实验室为学生提供Visual Studio的IDE环境。学生可使用该环境，进行集成的编辑、调试和运行。在VS环境中，重点了解如何创建项目、断点设置、跟踪调试、内存分析等手段。

对于使用Linux或Unix作为操作系统的学生，建议使用Emacs、Vim等工具作为编辑工具，使用gcc作为编译工具，使用gdb作为调试工具。以上工具学生自行通过网络下载。

### 练习3

该练习帮助学生理解C源代码到执行代码的转换过程，通过对不同代码阶段的观察，了解代码的表现形态。

源代码（main.c）：

#include <stdio.h>

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

{

// insert code here...

printf("Goodbye, Dannis!\n");

return 0;

}

通过对main.c的预处理、编译、汇编、链接等过程，了解不同形态下main.i、main.s、mian.o等的表现形态。通过搜索stdio.h，查找其存储位置，并了解其源码的含义。

### 练习4

理解const、static等概念的正确使用。

源代码（或片段）：

1、赋值操作

char \*cp;

const char \*ccp;

ccp = cp;

//cp = ccp;

2、const使用

const int limit = 10;

const int \* limitp = &limit;

int i = 27;

limitp = &i;

//limit = 20;

3、赋值操作失败的原因

foo(const char \*\*p) { }

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

{

foo(argv);

}

4、整形提升

int array[] = { 23, 34, 12, 17, 204, 99, 16 };

#define TOTAL\_ELEMENTS (sizeof(array) /

sizeof(array[0]))

main()

{

int d= -1, x;

/\* ... \*/

if (d <= TOTAL\_ELEMENTS-2)

x = array[d+1];

/\* ... \*/

}

5、Fall Through

switch (2) {

case 1: printf("case 1 \n");

case 2: printf("case 2 \n");

case 3: printf("case 3 \n");

case 4: printf("case 4 \n");

default: printf("default \n");

}

**6、break的使用（以下代码仅为示例，学生需要仿照此方式自己书写代码调试）**

network code()

{

switch (line) {

case THING1: doit1(); break;

case THING2: if (x == STUFF) {

do\_first\_stuff();

if (y == OTHER\_STUFF)

break;

do\_later\_stuff();

} /\* coder meant to break to here...\*/

initialize\_modes\_pointer();

break;

default: processing();

} /\* ...but actually broke to here! \*/

use\_modes\_pointer();

/\* leaving the modes\_pointer uninitialized \*/

}

**7、staic的使用**

generate\_initializer(char \* string)

{

static char separator='';

printf( "%c %s \n", separator, string);

separator = ',';

}

### 练习5

**该练习要求学生通过对C语言中安全漏洞的测试，了解C语言的局限性。**

**源代码：**

**1、gets（）、fgets（）的使用（自行编写代码，测试其安全特性）**

**2、返回局部指针的示例，并给出5种不同的修改方案。**

/\* Convert the source file timestamp into a localized date string \*/

char \* localized\_time(char \* filename)

{

struct tm \*tm\_ptr;

struct stat stat\_block;

char buffer[120];

/\* get the sourcefile's timestamp in time\_t format \*/

stat(filename, &stat\_block);

/\* convert UNIX time\_t into a struct tm holding local time \*/

tm\_ptr = localtime(&stat\_block.st\_mtime);}

/\* convert the tm struct into a string in local format \*/ strftime(buffer, sizeof(buffer), "%a %b %e %T %Y", tm\_ptr);

return buffer;

}