《嵌入式 LINUX-C 语言编程》

实验手册

林世霖 编撰 粤嵌教育 校订

Version 3.0 (2016-03-29)

微信扫描二维码，每天领取一个技术福利！

目录

实验一：编程环境以及 C 编程入门

内容概要

一、LINUX C 编程环境
二、典型 C 程序实例概览
三、数据类型

范例剖析

范例一：一年大约有 $3.1536*10^7$s。编写一个程序，要求输入你的年龄，然后显示该年龄等于多少秒。

思路：假设你年龄为 N，那么该年龄合等秒数为 $s = N * 3.1536*10^7$，该程序关键点在于：我们要用合适的数据类型来表示这些量，由于数值比较大，因此不能用整型来表示，我们考虑用浮点数，代码如下：

```
// age2sec.c
#include <stdio.h>
#define    SEC_YEAR    3.1536e7

int main(void)
{
    short age;
    float seconds;
    printf("how old are you(between 0 -- 100): "); // age:0-100

    int ret1, ret2;

    while((ret1=scanf("%hd", &age)) != 1 || // input don't match the format
        (ret2=getchar()) != '\n' || // don't match the format
        age > 100 || age < 0)    // out of range
    {

        if(ret1 != 1 || ret2 != '\n')
        {
            while(getchar() != '\n'){;} // discard invalid inputs
            printf("input invalid!\n");
        }
        else
            printf("invalid age!\n");

        printf("how old are you(between 0 -- 100): "); // age:0-100
    }
    seconds = age * SEC_YEAR;
    printf("you have pass %.0lf seconds in your life!\n", seconds);
    return 0;
}
```

关注点：

1， 学习该范例的编程风格，缩进、空格、空行等。

2， 将 seconds 定义为 float 类型，而不是整型，这样才能装得下秒数。

3， while 循环条件是难点，其中包含了 3 个判断条件，ret1 必须等于 1，如果不等于 1 则表示输入格式不正确，ret2 必须等于'\n'，如果不等于'\n'则表示正确输入之后还带有错误输入，而 age 大于 CHAR_MAX-155 或者小于 CHAR_MIN 都将被视为无效的输入（年龄只能是 0-100），这三个条件中的任何一个不成立，都将导致 while 语句的循环，提示用户继续输入。

4， scanf("%hd", &age)中的%hd 表示要从键盘获得一个短整型数据，放到变量 age 中。h 表示 half，d 表示 decimal，即半个十进制整型（4 个字节），即短整型（2 个字节）

技术点强化

1、指出下列常量的类型和意义（如果有的话）：

a) '\b'

b) 1066

c) 99.44

d) 0XAA

e) 2.0e30

2、编写一个程序，实现如下功能：用户输入一个 ASCII 码值(如 66)，程序输出相应的字符。

3、Mr. Bing 写了下面这个程序，请指出你认为不妥的地方：

```
include "stdio.h"
main{ }
(
    float g; h;
    float tax, rate;
    g = e21;
    tax = rate * g;
    printf("%f\n", tax);
)
```

4、一个水分子的质量大约为 3.0*10-23g，1 夸脱水大约有 950g。编写一个程序，要求输入水的夸脱数，然后显示这么多水中包含多少个水分子。

5、假设 c 为 char 类型变量。使用转义序列、十进制值、八进制字符常量以及十六进制字符常量等方法将其赋值为回车符（使用 ASCII 码）。

6、说说'A'与"A"有什么区别？

7、有时候我们需要使用 uint32_t 类型变量代替 unsigned int 类型变量的原因是什么？

实验二：C 语言基本概念及格式化 IO 函数

内容概要：

一、字符串和格式化 IO
二、运算符，表达式和语句

范例剖析

范例二：编写一个程序，实现以下功能：用户输入一个分钟数，程序将其转换成以小时和分钟表示的时间并输出到屏幕上。（使用#define 来定义一个代表 60 的符号常量）

思路：假设用户输入了 N 分钟，因为 60 分钟=1 小时，因此 N 分钟包含的小时数应该是 N/60，只要是整数相除，得到的答案就是整小时数，而余下的分钟数，则是 N%60。代码如下：

```
//min2hour.c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <stdbool.h>
#include <limits.h>

#define    MIN    60

int main(void)
{
    int minutes;
    printf("minutes: ");

    while(1)
    {
        scanf("%d", &minutes); // no error check!
        if(minutes < 0)
        {
            fprintf(stderr, "input invalid, try again: ");
            continue;
        }
        break;
    }
    printf("%d hours and %d minutes\n", minutes/MIN, minutes%MIN); //  核心算法
    return 0;
}
```

技术点强化：

1、编写一个程序，此程序要求输入一个整数，然后打印出从输入的值(含)到比输入的值大 10(含)的所有整数值(比如输入 5，则输出 5 到 15)。要求在各个输出值之间用空格、制表符或者换行符分开。

2 、写出下面表达式运算后 a 的值，设原来 a=12。设 a 和 n 已定义为整型变量。

(1) a += a

(2) a -= 2

(3) a *= 2+3

(4) a /= a+a

(5) a %= (n%2)，n 的值为 5

(6) a += a -= a *= a

3、编写一个程序，该程序要求输入一个 float 型数并打印概述的立方值。使用你自己设计的函数来计算该值的立方并且将它的立方打印出来。main 函数负责把输入的值传递给该函数。

4、编写一个程序，此程序要求输入天数，然后将该值转换为星期数和天数。例如输入 18，则要求输出：

18 days are 2 weeks, 4days.

5、分析并解释以下程序的执行结果。

```
#include <stdio.h>
int main(void)
{
    int a,b,c,d;
    a=10;
    b=a++;
    c=++a;
    d=10*a++;
    printf(“b，c，d：%d，%d，%d”，b，c，d);
    return 0;
}
```

实验三：C 语言控制流

内容概要：

一、控制流
二、字符 IO 和输入确认

范例剖析

范例三：当用户输入 5 的时候，使用嵌套循环产生下列图案（5 行美元符号，每行递增一个字符）：

```
$
$$
$$$
$$$$
$$$$$
```

思路：要输出这个图案，先要决定输出多少行，这个由用户的输入决定，每一行输出若干个星号，经过分析可以发现，输出的星号数目跟行数相等。代码如下：

```
//dollars.c
#include <stdio.h>

int main(void)
{
    int num, i=0;
    scanf("%d", &num); // no errno check!

    while((num-i) > 0)
    {
        int j;
        for(j=0; j<i+1; j++)
        {
            printf("$");
        }
        printf("\n");
        i++;
    }
    return 0;
}
```

关注点：

1， 用 for 和 while 构成二重循环，while 循环控制输出的行数，for 控制每一行输出的星号数目。

2， 程序中的 scanf()函数没有出错处理。

技术点强化：

1、编写一个程序，要求用相应的控制流语句往屏幕打印 26 个小写字母。

2、编写一个程序，用户输入某个大写字母，产生一个金字塔图案。例如用户输入字母 E，则产生如下图案：

```
    A
   ABA
  ABCBA
 ABCDCBA
ABCDEDCBA
```

3、编写一个程序，该程序读取输入直到遇到#字符，然后报告读取的空格数目、读取的换行符数目以及读取的所有其他字符数目。

4、编写一个程序，接受一个整数输入，然后显示所有小于或等于该数的素数。

5、输入一个华氏温度，要求输出摄氏温度。要求结果保留 2 位小数。

转换公式为：c = 5(F-32)/9

6、打印如下图案：

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

7、将一个十进制数转换为十六进制数。比如输入 10，输出 0xA

实验四：C 语言函数

内容概要：

一、函数
二、字符串和字符串函数

范例剖析

范例四：编写一个函数，判断一个整数是否为素数。
思路：

素数是这样的数：只能被 1 和本身整除（1 既不是素数也不是合数），假设给定一个整数 N，只要用 2,3,4 …… N-1 来整除 N，如果都不能除尽，则 N 是素数。

当然算法可以更加简练一点，判断一个数是否素数，只需要从 2 开始尝试整除，一直尝试到 N 的开方即可，不需要算到 N-1。根据这个思路可以优化程序。

代码如下：

```
//prime_number.c
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>

bool prime(int num)
{
    int i;
    if(num < 0)
        num *= -1;
    for(i=2; i*i<=num; i++)
    {
        if(num%i == 0)
            return false;
        else
            continue;
    }
    return true;
}

int main(void)
{
    int num;
    printf("Input a number(between %d and %d):", INT_MIN+1, INT_MAX-1);

    scanf("%d", &num); // no errno check!

    if(num==0 || num==1 || !prime(num))
```

```
        printf("humm, %d is NOT a prime.\n", num);
    else
        printf("great, %d IS a prime!\n", num);

    return 0;
}
```

关注点：

1， 定义一个函数 bool prime(int number)，用它来判断一个整数是否素数，如果是返回真，否则返回假。

2， prime 函数的算法主要就是从 2 开始尝试整除，一直尝试到 N 的开方，但程序中没有调用库开方函数 sqrt()，而改用 i*i <= num 替换达到相同的效果，因为调用 sqrt 函数速度更慢。

3， 函数并没有处理 scanf()的错误情况，也没有考虑负整数的情况。

技术点强化：

1、 说明函数传参的方式和异同。

2、 写出下面所描述的各个函数的 ANSI 函数头。注意：只写出函数头即可，不需要实现。

a) donut()接受一个 int 类型的参数，然后输出若干个 0，输出 0 的数目等于参数的值。

b) gear()接受两个 int 类型的参数并返回 int 类型的值。

c) stuff_it()的参数包括一个 double 类型的值以及一个 double 类型变量的地址，功能是把第一个数值存放到指定的地址中。

3、 编写一个函数，使其返回 3 个整型参数中的最大值。

4、 编写一个函数 my_power，用循环的方法实现返回一个 float 类型数的某个整数次幂(保留 6 位小数)。例如：调用 my_power(3.14, -2)返回 0.101424

5、 编写一个程序，将两个字符串连接起来，不要用 strcat 或 strncat 函数。

6、 编写一个函数 Fibonacci()，要求程序输出第 n 项斐波那契数，n 由用户输入。

斐波那契数列：1, 1, 2, 3, 5, 8, 13, 21, … …

7、编写一个程序，清除用户输入字符串中的空格符并将之输出。（例如用户输入”a b”，输出”ab”）

实验五：数组与指针

内容概要：

一、数组与指针（1）

范例剖析

范例五：编写一个函数，把两个数组内的相应元素相加，结果存储到第 3 个数组内。也就是说，如果数组 1 具有值 2, 4, 6, 8，数组 2 具有值 1, 0, 3, 6，则函数对数组 3 赋值为 3, 4, 9, 14。（该函数的参数包括 3 个数组名和数组大小）

思路：

先定义三个数组，给两个数组里面放置一些数值，然后用一个函数将它们对应的元素加起来放到第三个数组中去。

注意，给一个函数传递数组的时候，编译器会自动将传递的数组处理成一个指向数组首元素的指针，因此必须用另外的参数传递数组边界。代码如下：

```
// summary.c
#include <stdio.h>
#define   LIM   4

void sumary(int array1[], int array2[], int array3[], int size)
{
    int i;
    for(i=0; i<size; i++)
        array3[i] = array1[i] + array2[i];
}

int main(void)
{

    int array1[LIM] = {2, 4, 6, 8};
    int array2[LIM] = {1, 0, 3, 6};
    int array3[LIM];

    sumary(array1, array2, array3, LIM);

    int i;
    for(i=0; i<LIM; i++)
        printf("%d\t", array3[i]);

    printf("\n");
    return 0;
}
```

关注点：

1， 定义了三个数组 array1，array2 和 array3，第一和第二个数组被初始化了，注意不能越界。
2， 将这三个数组传递给函数 summary 的时候，编译器将会自动将形参变成指向首元素的指针。

技术点强化：

1、假如有如下定义：

int a[3][5];

a. 用 1 种方法表示 a[2][3]的地址。
b. 用 2 种方法表示 a[2][0]的地址。
c. 用 3 种方法表示 a[0][0]的地址。

2、编写一个函数，返回一个 double 型数组中最大和最小值的差值，并在一个简单的程序中测试这个函数。

3、用变量 a 给出下面的定义

a) 一个整型数
b) 一个指向整型数的指针
c) 一个指向指针的的指针，它指向的指针是指向一个整型
d) 一个有 10 个整型数的数组
e) 一个有 10 个指针的数组，该指针是指向一个整型数的
f) 一个指向有 10 个整型数数组的指针
g) 一个指向函数的指针，该函数有一个整型参数并返回一个整型数\
h) 一个有 10 个指针的数组，该指针指向一个函数，该函数有一个整型参数并返回一个整型数

温馨提示（怎么阅读复杂声明）：
A) 从左到右，遇到的第一个标识符，就是要说明的主体。
B) 以这个主体为中心，剥洋葱式地去解释。

4、下面的程序将打印出什么？解释原因

```
#include <stdio.h>
int main(void)
{
    int ref[ ] = {8, 4, 0, 2};
    int *ptr;
    int index;
    for(index = 0, ptr = ref; index<4; index++, ptr++)
    printf(“%d %d\n”, ref[index], *ptr);
    return 0;
}
```

5、在上一题中，ref 是哪些数据的地址？ref+1 呢？++ref 指向什么？

6、下面每种情况中*ptr 和*(ptr+2)的值分别是什么？

a)

```
int *ptr;
int torf[2][2] = {12, 14, 16};
ptr = torf[0];
```

b)

```
int *ptr;
int fort[2][2] = {{12}, {14, 16}};
ptr = fort[0];
```

7、给定两个相同的整型数组，将他们的各个元素的值相加存放到另一个整型数组中。

8、假设有如下声明：

```
float apple[10],
float apple_tree[10][5],
float *pf,
float weight = 2.2;
int    i = 3;
```

则下列语句中那些是正确的，哪些是错误的？原因是什么？

a. apple[2] = weight;

b. scanf(“%f”, &apple); scanf(“%f”, &apple[0]); scanf(“%f”, apple);

c. apple = weight;

d. printf(“%f”, apple[3]);

e. apple_tree[4][4] = apple[3];

f. apple_tree[5] = apple;

g. pf = weight;

h. pf = apple;

实验六：数组与指针

内容概要：

一、数组与指针（2）

技术点强化：

1、以下代码中的两个 sizeof 用法有问题吗？

```
void upper_case(char str[ ])
{
    int   i;
    for(i = 0; i < sizeof(str) / sizeof(   str[0]   );   i++)
    for(i = 0; i < sizeof(str) / sizeof( *(str+0)   );   i++)
    for(i = 0; i < sizeof(str) / sizeof(    ‘a’    );    i++)
    {
        if(str[i] > ‘a’   &&   str[i] < ‘z’)
            str[i] -= (‘a’ - ‘A’);
    }
}
int main(void)
{
    char str[ ] = “aBcDe”;
    printf(“length of the string: %d\n”, sizeof(str) / sizeof(str[0]));
    upper_case(str);
}
```

2、在 x86 平台下，分析以下代码的输出结果：

```
#include <stdio.h>
int main(void)
{
    int a[4] = {1, 2, 3, 4};
    int *p1=(int *)(&a+1);
    int *p2=(int *)((int)a+1);
    printf("%x, %x\n", p1[-1], *p2);
    return 0;
}
```

3、声明一个二维 int 型数组 a，再声明另一个一维数组指针数组，使该数组的每一个指针分别指向二维数组中的每一个元素(即每一个一维数组)，然后利用数组 b 计算数组 a 的和。

4、一个有 N 个元素的整型数组，求该数组的各个子数组中，子数组之和的最大值是多少？
例如数组 a[6] = {-2, 5, 3, -6, 4, -8, 6};则子数组之和的最大值是 8 (即 a[1] + a[2])。

5、编写一个程序，初始化一个 3x5 的二维 double 型数组，并利用一个基于变长数组的函数把该函数赋值到另一个二维数组，另外再写一个基于变长数组的函数来显示两个数组的内容。这两个函数应该能够处理任意的 NxM 数组。

6、编写一个程序，去掉给定字符串中重复的字符。例如给定”google”，输出”gole”。（华为笔试题）

实验七：LINUX-C 进程内存布局

内容概要：

一、存储类，链接和内存管理
二、LINUX C 内存映像

范例剖析

范例六：指出以下代码第二次输出结果，解释原因。

```
void other(void);
int main(void)
{
    extern  int  a;
    int  b=0;
    static int  c;
    a += 3;
    func ();
    b += 3;
    func ();
}
int  a=5;
void func(void)
{
    int  b=3;
    static  int  c=2;
    a += 5;
    b += 5;
    c += 5;
    printf("%d, %d, %d\n", a, b, c);
    c=b;
}
```

关注点：

1， 变量 a 是全局变量，因此在 main 函数和在 func 函数中访问的 a 都是同一个 a，任何地方对 a 的操作都会造成影响。

2， 变量 b 在 main 函数和 func 里面都是普通的局部变量，局部变量又叫临时变量，在进入函数体的时候，由系统在栈中临时分配内存，退出函数的时候又临时释放内存。因此 main 函数前后两次调用 func 时分别产生了两个 b，而 main 函数中也有一个 b，这三个变量没有任何关系。

3， 变量 c 是静态局部变量，存储在数据段中，而不是在栈中，因此在多次调用 func 函数的过程当中，访问的都是同一个 c。在 main 函数中也有一个这样的静态局部变量 c，这个跟 func 中的 c 也没有任何关系。

技术点强化：

1、 哪一存储类的变量在包含他们的程序运行时期内一直存在？哪一存储类的变量可以在多个文件中使用？

哪一存储类的变量只限于在一个文件中使用？

2、 代码块作用域变量具有哪种链接类型？

3、 说出 C 程序中所有不同的存储类变量在内存中的详细分布情况。

4、 编写一个函数，它返回函数自身被调用的次数，并在一个循环中测试之。

5、分析以下代码的输出结果并解释原因。

```
void    get_memory(char    **p)
{
      *p = "hello world“;
}
void Test(void)
{
      char *str = NULL;
      get_memory(&str);
      printf(“%s\n”,    str);
}
```

实验八：结构体等组合数据类型

内容概要：

一、复杂声明
二、结构体、共用体和枚举

范例剖析

范例七：假设有以下结构：

```
struct    gas
{
    float    distance;
    float    gals;
    float    mpg;    // mpg = distance * gals
};
```

a) 设计一个函数，它接受一个 struct gas 参数。假定传递进来的结构包括 distance 和 gals 信息。函数为 mpg 成员正确计算初值并返回这个完整的结构。

b) 设计一个函数，它接受一个 struct gas 参数的地址。假定传递进来的结构包括 distance 和 gals 信息。函数为 mpg 成员正确计算初值并把它赋给恰当的成员。

思路：

题目已经给出了结构体的模板，只需要在程序中定义该种类型的结构体即可。结构体可以像普通变量一样作为函数参数传递，也可以作为函数返回值返回给调用者。代码如下：

```
//gas.c
#include <stdio.h>

struct gas
{
    float distance;
    float gals;
    float mpg; // mpg = distance * gals
};

//pass the whole structure to this function
struct gas    deal_with_struct(struct gas bill)
{
    bill.mpg = bill.distance * bill.gals;
    return bill;
}

//pass the pointer to this function
void deal_with_pointer(struct gas *pbill)
{
```

```
    pbill->mpg = pbill->distance * pbill->gals;
}

int main(void)
{
    struct gas my_bill;
    my_bill.distance = 1.2;
    my_bill.gals = 3.4;

    my_bill = deal_with_struct(my_bill);
    printf("my bill's mpg: %f\n", my_bill.mpg);

    struct gas your_bill;
    your_bill.distance = 1.2;
    your_bill.gals = 3.4;

    deal_with_pointer(&your_bill);
    printf("your bill's mpg: %f\n", your_bill.mpg);
    return 0;
}
```

关键点：

1， 在 main 函数中定义了一个结构体变量 my_bill，然后将这整个结构体传递给 deal_with_struct 函数，这是按值传递，在 deal_with_struct 函数中的 bill 是 my_bill 的拷贝，计算完数值之后，又通过函数返回值返回给 main 函数中的 my_bill。

2， 在 main 函数中定义的另一个结构体变量 your_bill，则通过将自己的地址传递给 deal_with_pointer 函数来计算，这样传递节省了系统资源，效率更高。

技术点强化：

1、指出以下代码片段的不妥之处？

```
struct node
{
    char itable;
    int num[20];
    char * togs;
};
int x;
```

2、设计一个结构模板，保存一个月份名、一个 3 个字母的该月份的缩写、该月的天数，以及月份号。

```
struct month
{
    char *name;
    char brief_name[4];
    int days;
    unsigned int n;
```

```
};

struct month a;
a.name = “September”;
strcpy(a.brief_name, “Sep”);
```

3、分析以下结构所占的存储空间大小：

```
struct animals
{
    char dog;
    unsigned long cat;
    unsigned short pig;
    char fox;
};
```

4、定义一个结构体变量(包括年月日)。计算该日在本年中是第几天？注意闰年问题。

5、 声明一个枚举类型，使用 choices 作为标记，将枚举常量 no、yes 和 maybe 分别设置为 0、1 和 2。

6、 声明 4 个函数，并把一个指针数组初始化为指向它们。每个函数接受两个 double 参数并返回 double 值。

7、假设有以下说明和定义：

```
typedef     union
{
    long i;
    int k[5];
    char c;
} fruit;

struct     creature
{
    short cat;
    fruit apple;
    double dog;
};
fruit berry;
```

则语句 printf("%d", sizeof(struct creature)+sizeof(berry));的执行结果是？

9、编写一个 transform()函数，它接受 4 个参数：包含 double 类型数据的源数组名，double 类型的目标数组名，表示数组元素个数的 int 变量以及一个函数名(或者等价的指向函数的指针)。transform()函数把指定的函数作用于源数组的每个元素，并将返回值放到目标数组中。

例如：transform(source, target, 100, sin);
这个函数调用 sin(source[0])赋给 target[0]，等等。共有 100 个元素。在一个程序中测试该函数，调用 4 次 transform()，分别使用 math.h 函数库中的两个函数以及自己设计的两个适合的函数作为参数。

实验九：高级议题

内容概要：

一、预处理（头文件、复杂宏、条件编译等）
二、可移植性
三、变参函数、递归函数和回调函数

技术点强化：

1、 写一个带参数的宏 MIN(x, y)，这个宏输入两个参数并返回较小的一个。

2、 用预处理指令#define 声明一个常数，用以表明 1 年中有多少秒（忽略闰年问题）。

3、 某头文件中有以下语句，解释其作用：

```
#ifndef   SOME_HEADER_H_
#define SOME_HEADER_H_
… …
#endif
```

4、 设计一个 C 函数，若处理器是大端序的则返回 0，若处理器是小端序的则返回 1。

5、 编写一个函数，计算 1+2+3+4+…+n 的值。

6、 用递归的思想重做实验四的第 4 道题（my_power 函数）。

7、下面函数实现数组元素的逆转，请填写空白处使其完整。

```
void   recur(int   a[ ],   int   k)
{
    int   tmp;
    if( _____ )
    {
        recur( _____,  _____ );
        tmp = a[0];
        a[0] = a[k-1];
        a[k-1] = tmp;
    }
}
```

部分题目详解

题目 1：编写一个程序，实现如下功能：用户输入一个 ASCII 码值(如 66)，程序输出相应的字符。
示例代码 1：

```c
#include <stdio.h>

int main(void)
{
        char ch;

        scanf("%hhd", &ch);
        printf("%c\n", ch);

        return 0;
}
```

ascii2character1.c

专家剖析：

为了可以读取用户从键盘输入的整数，以及可以向屏幕输出结果，我们使用了标准库函数 scanf()和 printf()，scanf(“%hhd”, &ch)从键盘接受一个整数，并将其放进单字节变量 ch 中，然后 printf(“%c\n”, ch)将这个整数以字符形式打印到屏幕。但是这个代码没有任何错误检测，当用户输入有误，比如输入范围超过 0-127，或者输入的不是数字，程序就不能正常运行。

示例代码 2：

```c
#include <stdio.h>

int main(void)
{
        short ch;

        printf("pls input an ASCII value(0-127): ");
        int ret1, ret2;

        /*************************************
           deal with all invalid conditions:
           1: input doesn't match the format.
           2: input out of range.
        *************************************/
        while((ret1=scanf("%hd", &ch)) != 1 ||
                ((ret2=getchar()) != '\n') ||
                (ch>CHAR_MAX || ch<0)){

                if(ret1!=1 || ret2!='\n')
                        while(getchar() != '\n'); //discards invalid inputs

                printf("input invalid!\n");
                printf("pls input an ASCII value(0-127): ");
        }

        printf("the character of %hd is '%c'\n", ch, (char)ch);

        return 0;
}
```

ascii2character2.c

专家剖析：

为了能够检测用户的输入是否超过 0-127 的范围，第 5 行定义了一个 short 型变量 ch 来存储用户的输入，这样，当用户输入超过 127 但不超过 short 的最大值的时候，本程序可以正常工作。

另外，ret1 不为 1 时或者 ret2 不为'\n'时都代表用户输入了非法字符，第 20 行的 while 循环用来清空用户非法输入的字符，以保证下次能正常接收输入操作。

最后，该程序还是有 BUG 的，short 类型的变量 ch 并不能保证用户的输入一定是有效的，因为 short 类型的最大值是 32767（2^{15}-1），再加 1 就变成最小值-32768（-2^{15}），从（-2^{15}~2^{15}-1）是 short 类型能处理掉的范围，所以当用户输入 65536 时，我们的程序就无能为力了，它会以为用户输入了 0 而觉察不出来。

示例代码 3：

```
 1 #include <stdio.h>
 2 #include <stdbool.h>
 3 #include <string.h>
 4 #include <stdlib.h>
 5 #define SIZE 5
 6 short calculate(char input[])
 7 {
 8         short num=0;
 9         int len;
10         len = strlen(input)-1;
11         int pow = 1, i = 1;
12         while(len-i > -1)
13         {
14                 num += (input[len-(i++)]-'0')*pow;
15                 pow *= 10;
16         }
17         return num;
18 }
19 bool is_digit(const char *input)
20 {
21         bool flag = true;
22         int i=0, len=strlen(input)-1;
23         while(i<len)
24         {
25                 if(input[i]>'9' || input[i]<'0')
26                 {
27                         flag = false;
28                         break;
29                 }
30                 i++;
31         }
32         return flag;
33 }
34 bool more_then(const char input[], int limit)
35 {
36         return(input[limit-1] != '\0' && input[limit-1] != '\n');
37 }
```

```
38 int main(void)
39 {
40         short num;
41         static char input[SIZE];
42         while(1)
43         {
44                 memset(input, 0, SIZE);
45                 printf("pls input an ASCII value(0-127): ");
46                 fgets(input, SIZE, stdin);
47                 if(input[0] == '\n')    //1. empty line
48                         continue;
49                 if(more_then(input, 4)) //2. out of range
50                 {
51                         printf("invalid input!\n");
52                         while(getchar() != '\n');
53                         continue;
54                 }
55                 if(!is_digit(input))    //3. NOT digits only
56                 {
57                         printf("invalid input!\n");
58                         continue;
59                 }
60                 num = calculate(input);
61                 if(num > 127 || num < 0) //4. out of range
62                 {
63                         printf("out of range!\n");
64                         continue;
65                 }
66                 /*****************************************
67                 Congratulations! no logical flaw was found,
68                 now break out of the loop and print the answer.
69                 *****************************************/
70                 break;
71         }
72         printf("character: '%c'\n", num);
73         return 0;
74 }
```

ascii2character3.c

专家剖析：

示例代码 3 用字符的方式接收用户的输入（第 41 行的字符数组 input），这样才能彻底解决用户输入超过范围的问题。

第 47 行，判断当用户直接按回车时，让用户继续输入。

第 49 行，判断用户输入超过 4 个字符时，让用户继续输入。（用户最多可以输入 3 位数）

第 55 行，判断用户是否输入全数字，否则让用户继续输入。

第 60 行，将用户的输入转化为数字。

第 61 行，判断用户的输入是否在合理的范围之内。

题目 2：一个水分子的质量大约为 $3.0*10^{-23}$g，1 夸脱水大约有 950g。编写一个程序，要求输入水的夸脱数，然后显示这么多水中包含多少个水分子。

示例代码：

```c
#include <stdio.h>

#define WATER_MOLCULER 3.0e-23

int main(void)
{
        double water_molculers;
        float quota;
        printf("how many quarts of the water? ");

        if(scanf("%f", &quota)!=1 || quota<0)
        {
                printf("we need a positive digit. Bye-bye!\n");
                return 1;
        }

        water_molculers = (quota*950) / WATER_MOLCULER;

        printf("awesome! you've got %le water molecules!\n",
                                water_molculers);

        return 0;
}
```

water_molecules.c

专家剖析：

由题目可知，如果有 N 夸脱水，则总共有 N*950 克，而每个水分子的质量是 $3.0*10^{-23}$g，则 N*950 克水含有的水分子数目是 $N*950 / 3.0*10^{-23}$ 程序中注意使用恰当的数据类型来表示浮点数，以免溢出。

题目 3：编写一个程序，此程序要求输入一个整数，然后打印出从输入的值(含)到比输入的值大 10(含)的所有整数值(比如输入 5，则输出 5 到 15)。要求在各个输出值之间用空格、制表符或者换行符分开。

示例代码：

```c
#include <stdio.h>
#include <limits.h>

int main(void)
{
        int begin;
        printf("pls input an integer: ");

        scanf("%d", &begin);
        int i;
        for(i=0; i<=10; i++)
        {
                printf("%d\t", begin + i);
        }
        printf("\n");
        return 0;
}
```

numbers.c

专家剖析：

第 6 行，定义了一个局部 int 型变量 begin，用来存储用户的输入。在用 scanf()读取用户输入之前，第 7 行用 printf()函数打印了一句温馨提示。

注意到第 9 行是不带任何错误检测的，你如果想要完整地检测用户的输入合法性，必须像题目 1 的示例代码 3 那样编写你的代码。

第 11 行到第 14 行是一个 for 循环，控制循环的变量 i 从 0 开始变化，每次执行完花括号里面的语句之后执行 i++自增，一直到 i 为 11 退出循环。

题目 4：编写一个程序，该程序要求输入一个 float 型数并打印概述的立方值。使用你自己设计的函数来计算该值的立方并且将它的立方打印出来。main 函数负责把输入的值传递给该函数。

示例代码：

```
1  #include <stdio.h>
2
3  double cube(float f)
4  {
5          return f*f*f;
6  }
7
8  int main(void)
9  {
10         float f;
11         printf("pls input a float num: ");
12
13         while(scanf("%f", &f) != 1 || // input don't match the format
14                         getchar() != '\n'){ // invalid input
15
16                 while(getchar() != '\n'); // discards invalid inputs
17
18                 printf("input error!\n");
19                 printf("pls input a float num: ");
20         }
21
22         printf("cube of %f is %lf\n", f, cube(f));
23         return 0;
24 }
```

cube.c

专家剖析：

从 main 函数看起，第 10 行定义了一个 float 类型的变量 f，第 13 行的 scanf 用来获取用户的输入。13 行的 while 循环语句对用户的输入做了一定的错误检测（但并不完整，当用户输入的数据过大溢出时该程序就不能正常运行）。

第 22 行的 printf()函数里面，第三个参数直接调用了自定义的函数 cube()，并且将变量 f 传递了给她。函数 cube()的功能就是计算 f 的立方值，此时程序将会跳转到第 3 行运行，将 f 的立方返回到第 22 行作为 printf 函数的第三个参数，被打印出来。

题目 5：编写一个程序，要求用相应的控制流语句往屏幕打印 26 个小写字母。

示例代码：

```c
#include <stdio.h>

int main(void)
{
        char alp;

        for(alp='a'; alp<='z'; alp++)
        {
                printf("%c", alp);
        }
        printf("\n");

        return 0;
}
```

alphabet.c

专家剖析：

第 5 行定义了一个 char 类型的变量 alp，用来存储字符。

第 7 行到第 10 行是一个 for 循环，变量 alp 从’a’（即 97）开始，每次执行外循环体语句（即 printf 语句）之后执行 alp++，注意，字符就是整型，是一种特殊的整型，特殊在只有一个字节，因此整型支持的运算字符都是支持的，比如自加符++。 一直循环到 alp 超过了’z’字符的 ASCII 码值的范围，退出 for 循环。

第 11 行的反斜杠是要打印一个换行符，使得打印出来的效果更好看。

题目 6：编写一个程序，用户输入某个大写字母，产生一个金字塔图案。例如用户输入字母 E，则产生如下图案：

```
    A
   ABA
  ABCBA
 ABCDCBA
ABCDEDCBA
```

示例代码：

```c
#include <stdio.h>
#include "myhead.h"

int main(void)
{
        printf("Pls input a latter: ");
        char ch;
        scanf("%c", &ch);

        if((ch<'A') || (ch>'Z'))
        {
                printf("we need a capital latter.\n");
                return 1;
        }

```

```
16          char  line;
17          line = ch - 'A' + 1;
18
19          int i, j;
20          for(i=1; i<=line; i++)
21          {
22                  // print blank space
23                  for(j=0; j<line-i; j++)
24                  {
25                          printf(" ");
26                  }
27
28                  // print ascending letters
29                  for(j=0; j<i; j++)
30                  {
31                          printf("%c", 'A'+j);
32                  }
33
34                  // print descending letters
35                  for(j-=2; j>=0; j--)
36                  {
37                          printf("%c", 'A'+j);
38                  }
39                  printf("\n");
40          }
41          return 0;
42  }
```

pyramid.c

专家剖析：

第 8 行到第 14 行，接受用户的输入，并检测确保输入的是大写字母。

第 16 行定义了一个变量 line，用来表示根据用户的输入，即将需要打印的金字塔的总行数。

第 20 行是一个大的 for 循环，用来控制需要打印的行数，显然需要打印的行数取决于用户的输入，在这里，我们需要循环 line 次，打印 line 行。

第 22 到 26 行，打印空格，经过分析得知，每一行所需的空格数目是 line - i 个。

第 28 到 32 行，打印升序字母，经过分析得知，第 i 行需要打印 i 个升序字母，从‘A’开始。

第 34 到 38 行，打印降序字母，经过分析得知，第 i 行需要打印 i – 1 个字母。

题目 7：编写一个程序，该程序读取输入直到遇到#字符，然后报告读取的空格数目、读取的换行符数目以及读取的所有其他字符数目。

示例代码：

```c
#include <stdio.h>
#include <ctype.h>

#define STOP '#'

int main(void)
{
        char c;
        int n_spaces = 0;
        int n_lines = 0;
        int n_characters = 0;

        printf("Enter text to be analyzed(# to terminate):\n");

        while((c=getchar()) != STOP)
        {
                switch(c)
                {
                        case ' ':
                                n_spaces++;
                                break;
                        case '\n':
                                n_lines++;
                                break;
                        default:
                                n_characters++;
                }
        }

        printf("spaces = %d, lines = %d, characters = %d\n", \
                        n_spaces, n_lines, n_characters);
        return 0;
}
```

counter.c

专家剖析：

这个程序主要关注一下关键字 switch 的用法。

第 15 行，用 getchar()来从标准输入设备（即键盘）获得用户输入的字符，只要不等于 STOP，循环就不断进行。得到的用户输入放在变量 c 当中，接下来用 switch 语句对其进行判断。

第 17 行判断 c 的值，第 19 行，如果 c 的值等于‘ ’（即空格），则 n_spaces 加 1.

第 22 行，如果 c 的值等于‘\n’（即换行），则 n_line 加 1.

否则，第 26 行，如果 c 的值是默认值（即其他普通字符），则 n_characters 加 1.

最后，将他们都打印出来。

题目 8：编写一个程序，接受一个整数输入，然后显示所有小于或等于该数的素数。

示例代码：

```c
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>

bool prime(int num)
{
        int i;
        for(i=2; i*i<(num+1); i++)
        {
                if(num%i == 0)
                        return false;
                else
                        continue;
        }
        return true;
}

int main(void)
{
        int boundary;
        printf("Input the boudary(between 0 and %d):", INT_MAX-1);

        int ret1, ret2;
        while((ret1=scanf("%d", &boundary)) != 1 || // input don't match the format
                        boundary<0 || boundary>INT_MAX-1 || // out of range
                        (ret2=getchar()) != '\n') // don't match the format
        {

                if(ret1!=1 || ret2!='\n') // discards invalid inputs
                        while(getchar() != '\n');

                printf("invalid input!\n");
                printf("Input the boudary(between 0 and %d):", INT_MAX-1);
        }

        int num;
        for(num=2; num < boundary+1; num++)
        {
                if(prime(num))
                        printf("%d\t", num);
        }
        printf("\n");

        return 0;
}
```

prime_numbers.c

专家剖析：

从 main 函数开始看起，第 24 行到 34 行检测用户输入。第 37 行是一个 for 循环，通过自定义函数 prime()来计算从 2 开始一直到 boundary 是否是素数，如果是，prime()函数返回真，让 printf 函数可以打印，否则返回假，不打印。

当程序执行到第 39 行的时候，将会跳转到第 5 行去运行，prime 这个函数需要判断传进来的数据 num 是否是素数，方法是将 num 依次整除从 2 开始到 num 的开方。开方运算可以使用库函数 sqrt()来实现，但是在循环体中不断调用这个复杂的库函数显然是不明智的，程序的第 8 行用了一种巧妙的等价的方法来达到这个目的，for 循环中的条件判断语句不是 i < sqrt(num)+ 1，而是 i * i < num + 1。时间节省了很多。

题目 9：输入一个华氏温度，要求输出摄氏温度。要求结果保留 2 位小数。转换公式为：c = 5(F-32)/9

示例代码：

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5         float fah, cel; //Fahrenheit and Celsius;
6         printf("pls input the temperature you wanna calculate: ");
7
8         while(scanf("%f", &fah) != 1 || getchar() != '\n')
9         {
10
11                 while(getchar() != '\n'); // discards invalid inputs
12
13                 printf("input invalid!\n");
14                 printf("pls input the temperature you wanna calculate: ");
15        }
16
17        cel = (5*(fah-32)) / 9;
18        printf("It equals %.2f\n", cel);
19        return 0;
20 }
```

F2C. c

专家剖析：

这个程序很简单，主要就是套用了一下摄氏度和华氏度的转换公式。程序中要注意，由于要转化的不是整数，因此将 fah 和 cel 定义成两个 float 类型的变量。

注意到第 18 行，printf 函数的格式控制符是%.2f，表示打印的时候保留小数点后两位。

题目 10：编写一个程序，用户输入一个整数，要求：1)求出它是几位数；2)分别打印出每一位数字；3)按逆序打印出各位数字。

示例代码：

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        int num, tmp;
        int digit;

        printf("pls input an integer: ");
        while(scanf("%d", &num) != 1 || getchar() != '\n')
        {
                while(getchar() != '\n');

                printf("invalid input!\n");
                printf("pls input an integer: ");
        }

        //digits
        int weight=10, i=1;
        tmp = abs(num); //absolute value of num
        while(1)
        {
                if((tmp-weight) < 0)
                {
```

```
25                      printf("numbers: %d\n", i);
26                      break;
27              }
28              weight *= 10;
29              i++;
30      }
31
32      //numbers
33      tmp = abs(num);
34      printf("digits: ");
35      while(i--)
36      {
37              weight /= 10;
38              digit = (tmp-(tmp % weight))/weight;
39              printf("%d ", digit);
40              tmp -= tmp-tmp%weight;
41      }
42
43      //invert
44      weight *= 10;
45      printf("\ninverted: ");
46      if(num < abs(num))
47              printf("-");
48      tmp = abs(num);
49      if(tmp == 0)
50              printf("0");
51      while(tmp)
52      {
53              printf("%d", tmp % weight);
54              tmp /= weight;
55      }
56      printf("\n");
57      return 0;
58 }
```

number_dealing.c

专家剖析：

从第 10 行到第 16 行，接受用户的输入，并且进行简单的输入检测。

从第 19 行到第 30 行，计算其位数。第 33 到 41 行，将其每一个数位输出。注意到，abs()函数是标准 C 库自带的数学函数，用来求一个数的绝对值（absolute value）。 第 44 到 55 行，用来倒序输出。

题目 11：打印如下图案：

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

示例代码：

```c
#include <stdio.h>

int main(void)
{
        int size;
        printf("pls input the size of the diamond(odd number): ");

        int ret1, ret2;
        while((ret1=scanf("%d", &size)) != 1 || // input don't match the format
                        size % 2 == 0 || size < 1 || // invalid input
                        (ret2=getchar()) != '\n') // don't match the format
        {
                if(ret1 != 1 || ret2 != '\n')
                        while(getchar() != '\n');
                printf("input invalid!\n");
                printf("pls input the size of the diamond(odd number): ");
        }
        int i=1, space, star;
        while(i<=((size/2)+1)) // upper half
        {
                for(space=0; space<((size/2)-i+1); space++)
                {
                        printf(" ");
                }
                for(star=0; star<((2*i)-1); star++)
                {
                        printf("*");
                }
                printf("\n");
                i++;
        }
        int j = 1;
        i -= 2;
        while(j<((size/2))+1) // lower half
        {
                for(space=0; space<(j++); space++)
                {
                        printf(" ");
                }
                for(star=0; star<((2*(i--))-1); star++)
                {
                        printf("*");
                }
                printf("\n");
        }
        return 0;
}
```

diamond.c

专家剖析：

这个程序将要打印一个钻石图案，将这个图案分成两部分，分别是上半部分和下半部分。两部分都是三角形，不同的是上半部分打印的是正立的，下半部分是倒立的。

从程序的第 19 行开始到第 31 行，打印正立三角形，19 行的 while 循环用来控制打印的行数，行数是整

个钻石图案的行数的一半加 1，即包括中间一行。里面的两个 for 循环用来打印空格和星号。

题目 12：编写一个函数，使其返回 3 个整型参数中的最大值。

示例代码：

```
1  #include <stdio.h>
2
3  int max(int x, int y, int z)
4  {
5          int ret;
6          ret = (x>y) ? x : y;
7          ret = (ret>z) ? ret : z;
8
9          return ret;
10 }
11
12 int main(void)
13 {
14         int a, b, c;
15         printf("pls input 3 integers: ");
16
17         while((scanf("%d%d%d", &a, &b, &c)) != 3 ||
18                         getchar() != '\n')
19         {
20                 while(getchar() != '\n');
21
22                 printf("invalid input!\n");
23                 printf("pls input 3 integers: ");
24         }
25
26         printf("the max value is: %d\n", max(a, b, c));
27         return 0;
28 }
```

max_value.c

专家剖析：

求三个数的最大值，本程序使用了一个叫做 max 的函数来实现。这个函数接受三个整型变量，利用条件运算符求出结果。第 6 行中，ret 保存了 x 和 y 的最大值。第 7 行，使得刚才求得的 ret 再与 z 比较，得到三个数的最大值。

注意：函数的调用在第 26 行，这行中的 a，b 和 c 被称为实参，即 arguments，函数调用将会使得程序的运行跳转到其相应的定义的位置，即第 3 行。这行中的 x，y 和 z 称之为形参，即 parameters，它们跟跟实参一一对应，形参用实参的值来初始化，但是它们是相互独立的，即各自占用不同的内存。

题目 13：编写一个函数 Fibonacci()，要求程序输出第 n 项斐波那契数，n 由用户输入。

斐波那契数列：1, 1, 2, 3, 5, 8, 13, 21

示例代码：

```c
#include <stdio.h>

int fibonacci(int num)
{
        if(num <= 0)
                return 0;
        else if(num == 1)
                return 1;
        else
                return (fibonacci(num-1) + fibonacci(num-2));
}

int main(void)
{
        int num;
        printf("which Fibonacci num do you want: ");

        int ret1, ret2;
        while((ret1=scanf("%d", &num)) != 1 ||
                        num < 0 ||
                        (ret2=getchar()) != '\n')

        {
                if(ret1 != 1 || ret2 != '\n')
                        while(getchar() != '\n');

                printf("invalid input!\n");
                printf("which Fibonacci num do you want: ");
        }

        printf("the %dth Fibonacci number is: %d\n", \
                                num, fibonacci(num));
        return 0;
}
```

fibonacci.c

专家剖析：

斐波那契数列，此程序用了一个递归函数来实现它。根据斐波那契数列的公式 F(n) = F(n-1) +F(n-2)对应的 C 程序代码就是第 10 行。

写递归函数时，要注意的一个问题是，必须有可以满足的条件使得程序终止递归。比如此例，当形参 num 小于等于 1 的时候，函数 fibonacci 可以直接返回，而不是无穷递归。

递归函数需要注意的另一个问题是，并非所有的适合使用递归算法的问题都可以使用递归函数，因为递归函数的效率是非常低的，需要的栈空间也会随着递归的加深而增大，函数调用的开销也很大，所以递归函数仅适用于非程序热点的地方。（程序的热点指的指制约程序性能的瓶颈部分）

题目 14：声明一个二维 int 型数组 arr，再声明另一个一维数组指针数组，使该数组的每一个指针分别指向二维数组中的每一个元素(即每一个一维数组)，然后利用数组 p2arr 计算数组 arr 的和。

示例代码：

```c
#include <stdio.h>

int main(void)
{
        int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};
        int (*p2arr[2])[3];

        p2arr[0] = &arr[0];
        p2arr[1] = &arr[1];

        int i, j, sum=0;
        for(i=0; i<2; ++i)
        {
                for(j=0; j<3; ++j)
                {
                        sum += (*p2arr[i])[j];
                }
        }

        printf("sum: %d\n", sum);
        return 0;
}
```

sum.c

专家剖析：

这道题考查对基本概念的精通程度。

第 5 行，定义了一个二维数组 arr，这个二维数组其实是由两个具有三个整型元素的一维数组 arr[0]和 arr[1]组成的一维数组。

第 6 行定义了一个数组，叫做 p2arr，这个数组有两个元素，这两个元素都是指针，这两个指针都指向具有三个元素的整型一维数组。所以，p2arr 是一个装了两个指向数组的指针的数组，简称数组指针数组。

由于 p2arr 的元素是用来指向具有三个元素的整型一维数组的指针，而刚好 arr 这个数组的元素就是这样的东西，很自然，我们可以将每一个 arr 数组的元素的地址，赋值给 p2arr 的元素。即第 8 行和第 9 行代码。

解释一下第 16 行，首先是取 p2arr 的元素，即里面的数组指针，然后对其进行解引用，那也就是所指向的数组，然后在对数组进行索引取值，赋值并累计给 sum。