# 第二次实验报告

实验一览

1、 用程序熟悉和掌握if语句的各种形式

2、 用程序熟悉和掌握switch语句，熟悉switch语句中break的作用。用一个实例采用if和switch语句实现，比较两个程序的可读性

3、 用一个实例说明while,for, do..while语句使用的区别和特点。

4、 用程序说明循环语句中break和continue的差异

5、 用实例说明多重循环中退出机制

6、给一个不多于5位的的正整数，要求：求出它是几位数，分别打印出每一位数字，最后按照逆序打印各位数字，例如原数为321，应输出为123。

7、 编写一猜数游戏程序，随机产生某个整数，从键盘反复输入整数进行猜数，当未猜中时，提示输入过大或过小.猜中时，指出猜的次数.最多允许猜20次.

9、.猴子吃桃问题：第一天吃掉总数的一半多一个，第二天又将剩下的桃子吃掉一半多一个，以后每天吃掉前一天剩下的一半多一个，到第十天准备吃的时候见只剩下一个桃子，求第一天开始吃的时候桃子的总数。

### 实验一

用程序熟悉和掌握if语句的各种形式

## 使用的程序

//compare the size of four numbers

#include<stdio.h>

void main()

{

int a,b,c,d;

int min.mid1,mid2,max;

int t;

scanf(“%d %d %d %d”,&a,&b,&c,&d);

if(a>b)

{

max=a;

min=b;

}else

{

max=b;

min=a;

}

if(c>d)

{

mid1=d;

mid2=c;

}else

{

mid1=c;

mid2=d;

}

if(min>mid1)

{

t=min;

min=mid1;

mid1=t;

}

if(max<mid2)

{

t=max;

max=mid2;

mid2=t;

}

if(mid1>mid2)

{

t=mid1;

mid1=mid2;

mid2=t;

}

printf(“%d<%d<%d<%d\n”,min,mid1,mid2,max);

}

## 实验步骤

1.打开BC，导入这个程序。

2.编码连接运行这个程序，输入测试数据，观察结果。

3.判断结果是否正确，若出错，则进行调试，体会if语句的用法。

## 输入的测试数据

第一组1 2 3 4 第二组1 3 2 4 第三组4 3 2 1 第四组3 4 1 2

## 实验截图



## 体会

if语句可以单独使用，可以配合else使用，更可以配合else if使用。当if内的语句只有一行时，可以不加大括号。注意千万不要在if(\*)后加分号。依靠if语句，程序不再是一条线走到底，非A则B，给程序带来了更多可能。

### 实验二

用程序熟悉和掌握switch语句，熟悉switch语句中break的作用。用一个实例采用if和switch语句实现，比较两个程序的可读性

## 使用的程序

//transform scores into ABC(if)

#include<stdio.h>

void main()

{

int scores;

char s;

scanf(“%d”,&scores);

if(scores<=100&&scores>=90)

{

s=’A’;

}else if(scores<90&&scores>=80)

{

s=’B’;

}else if(scores<80&&scores>=60)

{

s=’C’;

}else

{

s=0;

}

if(s==0)

{

printf(“You failed\n”);

}else

{

printf(“You get %c\n”,s);

}

}

//transform scores into ABC(switch)

#include<stdio.h>

void main()

{

int scores,sign;

char s;

scanf(“%d”,&scores);

sign=scores/10;

switch(sign)

{

case 10:

case 9:

s=’A’;

break;

case 8:

s=’B’;

break;

case 7:

case 6:

s=’C’;

break;

case 5:

case 4:

case 3:

case 2:

case 1:

case 0:

s=0;

break;

}

if(s==0)

{

printf(“You failed\n”);

}else

{printf(“You get %c\n”,s);

}

}

## 实验步骤

1.打开BC，分别导入这两个程序。

2.编码连接运行这个程序，输入测试数据，观察结果。

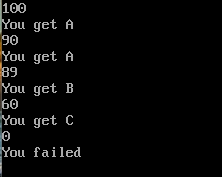
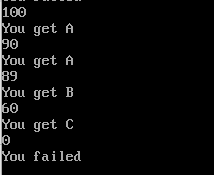
3.判断结果是否正确，若出错，则进行调试，体会break在switch中的作用。

4.分析if和switch的可读性。

## 输入的测试数据

第一组100 第二组90 第三组89 第四组 60 第五组0

## 实验截图

(if) (switch) 

## 体会

break可以终止switch语句的运行，如果没有break，则电脑运行完case n下面的语句后，还会继续运行switch内的其他语句，直至switch语句大括号内的内容被全部运行完。注意switch(\*)括号内的数据只能是int，在使用switch前，最好通过一些运算，来让case的种类少一些。

比较if和switch的可读性，对比实验使用的两个程序，虽然if的行数更少，但搭配else使用时，大括号很多，读起来不方便；而switch语句各个case整齐排列，确实更容易阅读。

### 实验三

用一个实例说明while,for, do..while语句使用的区别和特点。

## 使用的程序

#include<stdio.h>

void main()

{

int cntfor,cntwhile,cntdo;

int a,i=0;

scanf(“%d”,&a);

for(cntfor=1;cntfor<a;cntfor++)//for part

{

i+=cntfor;

}

printf(“answer by \’for\’ is %d, cntfor=%d\n”,i,cntfor);

cntwhile=1;//while part

i=0;

while(cntwhile<a)

{

cntwhile++;

i+=cntwhile;

}

printf(“answer by \’while\’ is %d, cntwhile=%d\n”,i,cntwhile);

cntdo=1;//dowhile part

i=0;

do

{

cntdo++;

i+=cntdo;

}while(cntdo<a);

printf(“answer by \’dowhile\’ is %d, cntdo=%d\n”,i,cntdo);

}

## 实验步骤

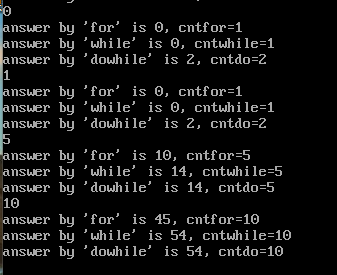
1.输入如下程序，编译运行。

2.分析for, while, do while部分输出值的区别。

## 输入的测试数据

第一组0 第二组1 第三组5 第四组10

## 实验截图



## 体会

较之于while和dowhile语句，for语句的整合性更好。此外，这三种程序循环判断的时机也不相同。以输入0时为例，for和while都是先判断是否进入循环（结果为不进入），dowhile是直接进入循环，一轮下来后再判断是否继续循环，这就导致了for循环0次，while循环0次，dowhile循环1次。最终输出结果的不同。当for，while，dowhile都能进入循环时（如输入5），输出结果不全相同，这是因为程序里for的cntfor++实际是在每轮循环的末尾，而while，dowhile中我把对应的cnt++放在了循环的开头。

### 实验四

用程序说明循环语句中break和continue的差异

## 使用的程序

#include<stdio.h>

void main()

{

int i;

int times=10;

for(i=0;i<times;i++)//continue part

{

if(i%2)

{

printf(“odd\n”);

continue;

}

printf(“even\n”);

}

printf(“i=%d\n”,i);

for(i=0;i<times;i++)//break part

{

if(i%2)

{

printf(“odd,break\n”);

break;

}

printf(“even\n”);

}

printf(“i=%d\n”,i);

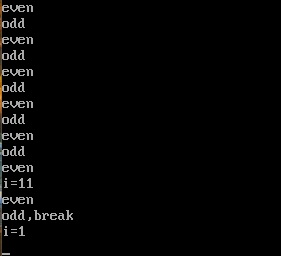
}

## 实验步骤

1.输入程序，编译连接运行。

2.分析continue部分和break部分的运行结果差别。

## 实验截图



## 体会

对for循环，break直接退出循环，continue会执行for的表达式3.**可以将for的流程理解为：for(表达式1;表达式2;表达式3).进入循环，执行表达式1，执行表达式2，若表达式2结果为1，则进入循环。执行循环内的语句，执行循环中的最后一条语句后，开始新一轮的循环，新一轮的循环先执行表达式3，再执行表达式2，若表达式2结果为1，则进入循环，若为0，则跳出循环……**

### 实验五

用实例说明多重循环中退出机制

## 使用的程序

#include<stdio.h>

void main()

{

int CNT,cnt;

int a=0;

for(CNT=0;CNT<10;CNT++)//break part

{

for(cnt=0;cnt<10;cnt++)

{

break;

}

printf(“break jump one time\n”);

a=1;

break;

}

if(a==0)

{

pirntf(“break can jump two times\n”);

}

while(CNT<10)//goto part

{

a+=CNT;

goto final;

}

if(1==0)

{

final:

printf(“goto arrives here”);

}

}

## 实验步骤

1.输入程序，编译连接运行。

2.分析多重循环的退出机制。

## 实验截图



## 体会

break在循环中使用时，每次只跳出一层循环，对for循环来讲，break后for的第三个表达式不会执行一次。goto也可以用于跳出循环，跳到哪里，用goto \*;在想抵达的地方写上\*:就行了。虽然方便，但在一定程度上破坏了程序的结构性，并不建议使用。

### 实验六

## 题目

给一个不多于5位的的正整数，要求：求出它是几位数，分别打印出每一位数字，最后按照逆序打印各位数字，例如原数为321，应输出为123。

## 设计思路

以输入532为例，先判断532/10是否为零，再判断532/100是否为零，再判断532/1000是否为零，以此确定数字的位数。逆序输出：532/100=5，532-100\*5=32，即消去最高位；类似操作，32/10=3,32-3\*10=2,消去第二高位，得到个位为2，输出个位。然后532-2=530，530/10=53，即消去个位，再对53实行类似的操作即可。

## 使用的程序

#include<stdio.h>

void main()

{

int number;

int figure=1;

int s=10;

int number2;

int cnt;

int cntt;

int s2=1;

int p;

scanf(“%d”,&number);

for(figure;number/s;figure++)

{

s\*=10;

}

printf(“the number is %d-figure\n”,figure);

printf(“%d\n”,number);

for(figure;figure>0;figure--)

{

cntt=figure;

number2=number;

for(cntt;cntt>1;cntt--)

{

for(cnt=cntt,s2=1;cnt>1;cnt--)

{

s2\*=10;

}

p=number2/s2;

number2-=p\*s2;

}

printf(“%d”,number2);

number-=number2;

number/=10;

}

printf(“\n”);

}

## 改进的程序（源自课上讲解）

#include<stdio.h>

void main()

{

long int number;

int figure;

int a,b;

scanf(“%d”,&number);

a=number/10;

for(figure=1;a!=0;figure++)

{

a/=10;

}

printf(“the number is %d-figure\n”,figure);

printf(“%d\n”,number);

for(figure;figure>0;figure--)

{

b=number%10;

printf(“%d”,b);

number-=b;

number/=10;

}

printf(“\n”);

}

## 实验步骤

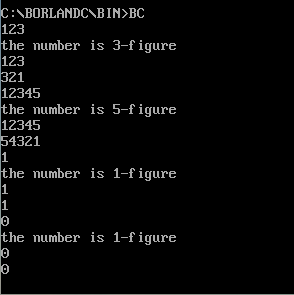
1.输入程序，编译连接运行。

2.输入测试数据，分析结果。

## 输入的测试数据

第一组123 第二组12345 第三组1 第四组0

## 实验截图



## 体会

注意C语言里^不是次方，要想次方，则需要用math.h中的pow(double,double)函数，但这个函数会把输入的两个量转换为double，可能不便于后续的运算。可以使用循环的方式计算次方。

这种排序的问题，在不利用数组的条件下确实非常复杂，有了数组，就可以把一个数的各位依次存储在数组中了。

**原程序的缺陷：1.number是int，范围有限，不能覆盖所有五位数。2.在逆序输出时过程过于复杂（%取余运算在数学中并不常见，这就导致了我在编写程序的时候忽略了这种方法）。**

### 实验七

## 题目

编写一猜数游戏程序，随机产生某个整数，从键盘反复输入整数进行猜数，当未猜中时，提示输入过大或过小.猜中时，指出猜的次数.最多允许猜20次.

## 使用的程序

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

void main()

{

long a;

long guess;

int cnt;

int i=1;

srand(time(NULL));

a=rand();

scanf(“%ld”,&guess);

for(cnt=1;a!=guess;cnt++)

{

if(guess>a)

{

printf(“Too large\n”);

}else

{

printf(“Too small\n”);

}

if(cnt==20)

{

i=0;

break;

}

scanf(“%d”,&guess);

}

if(i)

{

printf(“win! You try %d times”,cnt);

}else

{

printf(“Sorry your chances are out”);

}

}

## 改良方案（源于第三章ppt，例3.26）

随机数的生成部分

#include<time.h>

#include<stdlib.h>

……

randomize();

a=random(int);

## 设计思路

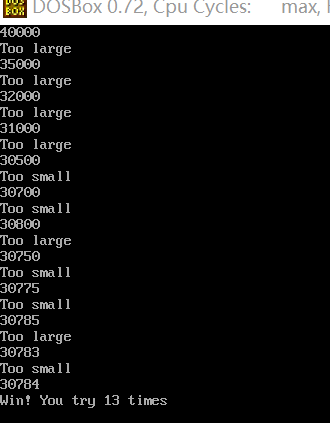
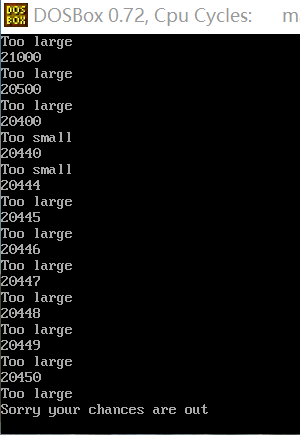
第一步，生成一个随机数。第二步，输入预测的数字。第三步，进入循环。第四步，用if判断是大还是小，给出反馈。第五步，若未猜中，则再次输入另一个预测的数字。

## 实验步骤

1.输入所用的程序,编译运行。

2.第一次认真猜，第二次乱猜。

## 实验截图

## 体会

如何产生一个随机数？我们可以利用rand()函数（需要事先调用stdlib.h函数库）。第一次实验时发现每次产生的随机数都完全相同，令我非常困惑。后来了解到rand()函数产生的随机数是伪随机数，是根据一个数值按照某个公式推算出来的，这个数值称之为“种子”,要想让每一次产生的随机数不一样，就需要每次提供给rand()函数的种子各不相同，然而”种子”在每次计算机开机后就被固定下来了，这就导致只用rand()函数，每次随机的数字完全相同。如何解决呢？可以用srand(unsigned int)函数重新”播种”，调用time.h函数库中的time(NULL)，即在产生随机数前先执行srand(time(NULL));进行重新播种，就可以让每次的种子不同，产生的随机数也就不同了。

**辨析随机数生成的几种方法**

**1.rand(),需要头文件stdlib.h,** **在调用此函数产生随机数前，必须先利用srand(int)设好随机数种子，如果未设随机数种子，rand()在调用时会自动设随机数种子为1。srand(int)也包含在头文件stdlib.h中。要想生成随机的种子，可以利用头文件time.h中的time(int).即例如srand(time(0))或srand(time(NULL))。**

**2.randomize(int)，需要头文件stdlib.h和time.h。这个函数的作用就相当于srand(time(NULL)),即初始化随机数的种子。**

**3.random(int),首先需要注意random函数并不是ANSIC标准，所以说，random函数不能在gcc,vc等编译器下编译通过。需要头文件stdlib.h。random(int num)会输出一个从0到num-1之间的随机数。**

### 实验八

## 题目

猴子吃桃问题：第一天吃掉总数的一半多一个，第二天又将剩下的桃子吃掉一半多一个，以后每天吃掉前一天剩下的一半多一个，到第十天准备吃的时候见只剩下一个桃子，求第一天开始吃的时候桃子的总数。

## 使用的程序

#include<stdio.h>

void main()

{

int peach;

int day;

for(peach=1,day=1;day<10;day++)

{

peach+=1;

peach\*=2;

}

printf(“%d”,peach);

}

## 实验步骤

1.输入程序，编译连接运行。

2.得到输出结果，判断结果是否正确。

## 实验截图



## 体会

此类循环问题，以此题为例，在设计循环结构的时候，先判断循环应该走几遍，此题第十天发现只剩一个桃子了，所以循环一共走9遍。在明确循环次数的前提下，设计循环变量day，若设day的初始值为1，循环走9遍就意味着day=1,2,3,4,5,6,7,8,9都满足循环条件，但10不满足，所以把循环进行的条件设为day<10.同时这类问题编写出的程序也不方便判断自己输出的答案对不对，可以把天数调小，比如把循环进行的条件改为day<3，手算的结果和程序输出的结果一致，就说明程序没有问题了。