一、do…while和while的区别

do…while，先执行再判断，一定会执行一次

while先判断再执行

```
实现死循环
while(1)
{

}

do
{

}while(1)
```

二、for循环

```
for(循环变量初始化;循环条件;修改循环变量)
{
    条件成立时，执行的代码;
}
```

执行逻辑：

1）先执行循环变量初始化

2）判断循环条件是否成立

3）条件成立时，执行{}内的代码

4）修改循环变量

```
int i = 0;  //不写循环变量初始化，在外面先定义出循环变量
for(;i<10;i++)
{
    printf("1");
}

//不写循环条件,在循环体内用if来时循环条件
for(int i = 0;;i++)
{
    if(i<10)
    {
        printf("1");
    }
}

//死循环
for(int i =0;;i++)
{
    printf("1");
}
```

```
//不写修改循环变量的语句
for(int i =0;i<10;)
{
    i++;
}

//三个都省略，死循环
for(;;)
{

}
```

循环的三要素：

1）循环变量初始值

2）循环条件

3）修改循环变量

三、goto

实际上是一个跳转，通过if来利用goto实现循环。

```
int i =0;
A:
printf("1");
if(i<10)
{
    goto A;
    i++;
}
```

```
#include <stdio.h>
int main(int argc, const char *argv[])
{
    int a = 0;
A:
    printf("1");
    if (a<10)
    {
        a++;  //改变a
        goto A;
    }
    return 0;
}
```

四、辅助控制关键字

4.1break

1）只能用在switch和循环中，不能放在goto中。

2）跳出离他最近的一层循环

4.2continue

1）只能用在循环中，不能放在switch和goto中

2）跳出本次循环

3）continue放在while循环中时，会跳出本次循环，修改循环变量的语句要放在continue前面，否则会死循环。

```c
#include <stdio.h>
int main(int argc, const char *argv[])
{
    int i =0;
    while (i<10)
    {
        i++;
        if (i<3)
        {
            continue;
        }
        printf("%d\n",i);
    }
    return 0;
}
```

```
ubuntu@ubuntu:~/22081/day6$ gcc 5.c
ubuntu@ubuntu:~/22081/day6$ ./a.out
0       0
0       1
0       2
0       3
0       4
ubuntu@ubuntu:~/22081/day6$ gcc 5.c
ubuntu@ubuntu:~/22081/day6$ gcc 5.c
ubuntu@ubuntu:~/22081/day6$ ./a.out
0       1       2       0
0       1       2       1
0       1       2       2
0       1       2       3
0       1       2       4
ubuntu@ubuntu:~/22081/day6$ gcc 5.c
ubuntu@ubuntu:~/22081/day6$ ./a.out
0       2       0
0       2       1
0       2       2
0       2       3
0       2       4
```

4.3return

1）退出当前函数，放在主函数中，退出整个程序。

4.4练习

用break完成，求[3,100]中的质数。

```
#include <stdio.h>
int main(int argc, const char *argv[])
{
    int i,j;
    for (i=3;i<=100;i++) //从3遍历到100
    {
        for (j=2;j<i;j++)//从2遍历到i-1
        {
            if (i%j==0) //判断i是否能被2---i-1中的数整除
            {
                break;  //如果能整除，不是质数
            }
        }
        if (i==j) //当内层循环正常退出时i与j应该相等，当用break退出时，i与j不等
        //判断内层循环是否是正常退出，正常退出说明没被整除是质数
        {
            printf("%d是质数\n",i);
        }
    }
    return 0;
}
```

五、数组

分为一维数组、二维数组和多维数组

5.1概念

能够连续存储多个相同的数据类型的元素的集合。

5.2一维数组

定义：

```
数据类型 数组名[数组的长度];
```

1）以数字，字母和下划线构成

2）不能以数字开头

3）不能和32个C语言关键字重名

4）严格区分大小写

初始化

```c
#include <stdio.h>
int main(int argc, const char *argv[])
{
    int arr[3] = {1,2,3}; //完全初始化
    int brr[3] = {1};  //不完全初始化，未初始化的部分默认为0
    int crr[]  = {1,2,3,4}; //未定义长度的初始化，编辑器就根据我们输入的数据给数组长度
    return 0;
}
```

[]:

在定义数组的时候，里面的数据表示的是数组的长度，在访问数组中元素的时候，表示的是数组的下标，

数组中元素的下标从0开始，最后一个元素的下标是（长度-1）

数组名的含义：

1）标识符

2）表示数组中首元素的地址。

通过数组下标访问

```c
#include <stdio.h>
int main(int argc, const char *argv[])
{
    int arr[3] = {1,2,3}; //完全初始化
    int brr[3] = {1};  //不完全初始化，未初始化的部分默认为0
    int crr[4] = {0};  //全部初始化为0，比较常用
    int drr[]  = {1,2,3,4};
    //int crr[] = {,,3};//不完全初始化的时候，只能省略后面的部分不能省略前面的部分。
    //未定义长度的初始化，编辑器就根据我们输入的数据给数组长度
    //arr = {1,2,3};//错误的，在初始化之后，不能再整体赋值了
    arr[0] = 2;
//  printf("%p\n",arr);
    printf("%d\n",arr[0]);
    printf("%d\n",arr[1]);
    //printf("%d\n",arr[100]); //数组越界不会报编译错误，数组越界访问的结果是不可预知的
    printf("%ld\n",sizeof(arr));
    printf("%ld\n",sizeof(drr)/sizeof(int));
    return 0;
}
```

数组的大小

存储的数据类型的长度*数据个数

sizeof(数组名);

数组的长度：

```
sizeof(数组名)/sizeof(数据中元素的类型)来求数组的长度。
```

```c
#include <stdio.h>
int main(int argc, const char *argv[])
{
```

```c
    int arr[6] = {0};
    int i;
    int max;
    int dex;
    for (i=0;i<6;i++)
    {
        scanf("%d",&arr[i]);
    }
    max = arr[0];
    dex = 0;
    for (i=0;i<6;i++)
    {
        if (arr[i]>max)
        {
            max = arr[i];
            dex = i;
        }
        //printf("%-3d%-3d",max,dex);
    }
    printf("%-3d%-3d",max,dex);
    return 0;
}
```