



§. 基础知识题 – 循环结构

要求:

- 1、完成本文档中所有的题目并写出分析、运行结果
- 2、无特殊说明，均使用VS2022编译即可
- 3、直接在本文件上作答，**写出答案/截图（不允许手写、手写拍照截图）**即可；填写答案时，为适应所填内容或贴图，**允许调整**页面的字体大小、颜色、文本框的位置等
 - ★ 贴图要有效部分即可，不需要全部内容
 - ★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可
 - ★ **不允许**手写在纸上，再拍照贴图
 - ★ **允许**在各种软件工具上完成（不含手写），再截图贴图
 - ★ 如果某题要求VS+Dev的，则如果两个编译器运行结果一致，贴VS的一张图即可，如果不一致，则两个图都要贴
- 4、转换为pdf后提交
- 5、**4月4日前**网上提交本次作业（在“文档作业”中提交）



§. 基础知识题 – 循环结构

贴图要求：只需要截取输出窗口中的有效部分即可，如果全部截取/截取过大，则视为无效贴图

例：无效贴图

```
Microsoft Visual Studio 调试控制台  
Hello, world!  
D:\Workspace\VS2019-Demo\Debug\cpp-demo.exe (进程 7484)已退出, 代码为 0。  
按任意键关闭此窗口. . .
```

例：有效贴图

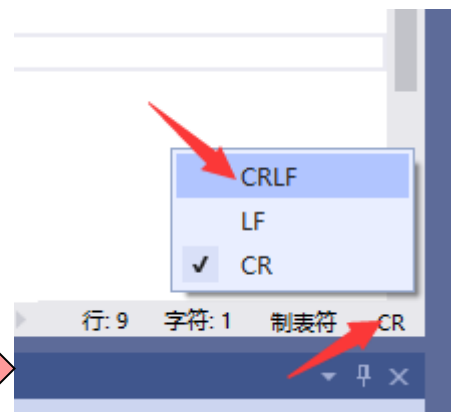
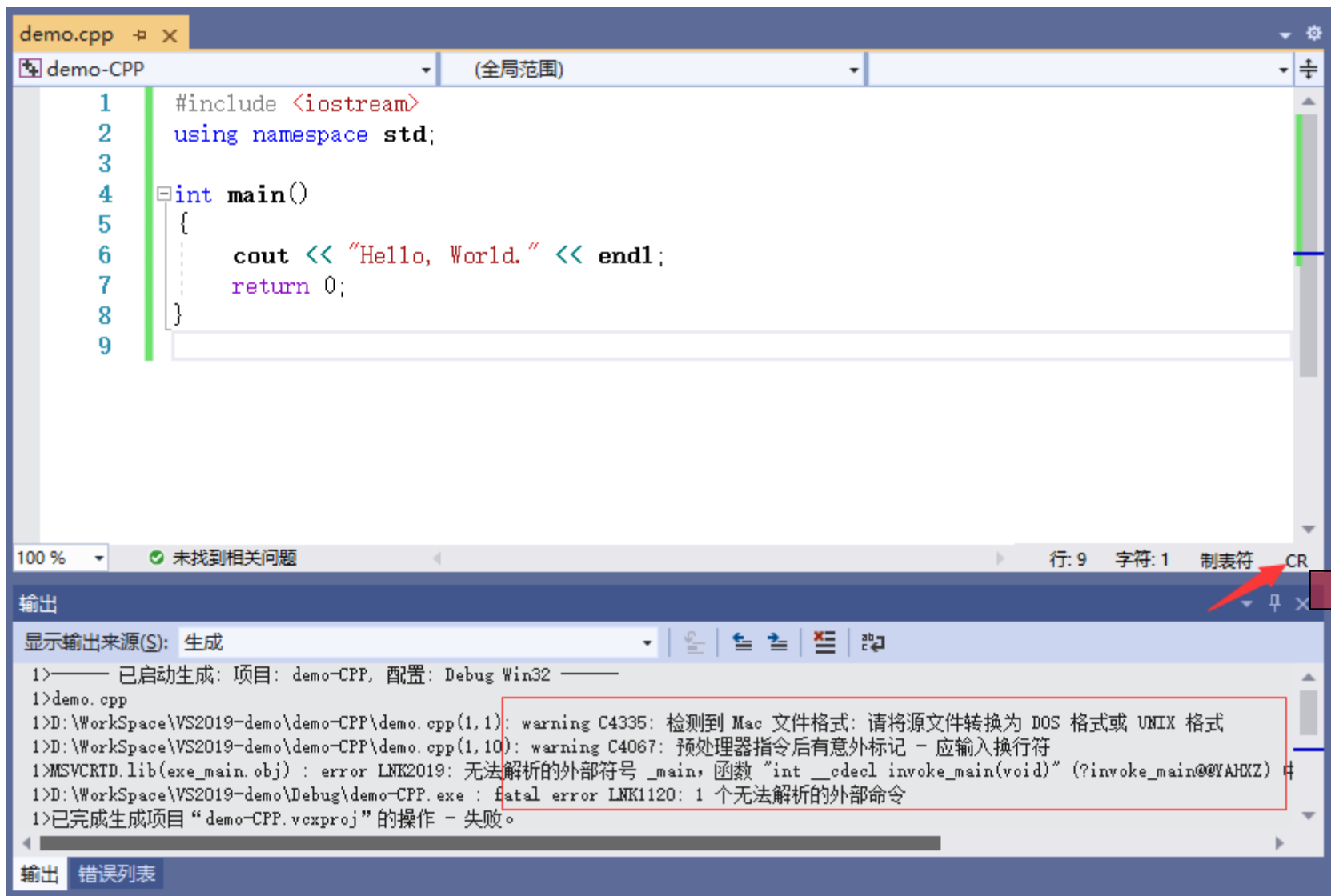
```
Microsoft Visual Studio 调试控制台  
Hello, world!
```



§. 基础知识题 - 循环结构

附：用WPS等其他第三方软件打开PPT，将代码复制到VS2022中后，如果出现类似下面的**编译报错**，则观察源程序编辑窗

的右下角是否为CR，如果是，单击CR，在弹出中选择CRLF，再次CTRL+F5运行即可



§. 基础知识题 - 关系运算、逻辑运算与选择结构



特别提示:

- ★ 本次作业的答案，除特别提示外，上课全讲过，课件上都有!!!
- ★ 作业本质就是对上课内容及课件的review(因为读懂程序的逻辑很重要)
- ★ 对上课接受程度较好的同学，可能有点重复/多余，但还得做



§. 基础知识题 – 循环结构

1、循环的嵌套

A. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <iostream>
using namespace std;

int main()
{
    int i, j, k;
    int count1 = 0, count2 = 0, count3 = 0;

    for(i=1; i<=100; i++) {
        ++count1;
        for(j=1; j<=100; j++) {
            ++count2;
            for(k=1; k<=100; k++)
                ++count3;
        }
    }

    cout << "count1=" << count1 << endl;
    cout << "count2=" << count2 << endl;
    cout << "count3=" << count3 << endl;
    return 0;
}
```

1、贴运行结果

```
count1=100
count2=10000
count3=1000000
```

2、当循环嵌套时，内层循环的执行次数和外层循环是什么关系？

当外层循环执行1次，内层循环执行到结束为止；之后外层循环执行第2次，再执行内层循环到结束为止；直到外层循环结束。

本题内层循环次数即为外层循环执行次数与内层循环单次执行次数之积。



§. 基础知识题 – 循环结构

1、循环的嵌套

B. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <iostream>
using namespace std;

int main()
{
    int i, j, k;
    int count1 = 0, count2 = 0, count3 = 0;

    for(i=1; i<=100; i++) {
        ++count1;
        for(j=i; j<=100; j++) {
            ++count2;
            for(k=j; k<=100; k++)
                ++count3;
        }
    }

    cout << "count1=" << count1 << endl;
    cout << "count2=" << count2 << endl;
    cout << "count3=" << count3 << endl;
    return 0;
}
```

1、贴运行结果

```
count1=100
count2=5050
count3=171700
```

2、当循环嵌套时，内层循环的执行次数和外层循环是什么关系？

当外层循环执行1次，内层循环执行到结束为止；之后外层循环执行第2次，再执行内层循环到结束为止；直到外层循环结束。

由于内层循环的初值与外层循环有关，内层循环次数是外层循环每一次执行次数的累计和。



§. 基础知识题 – 循环结构

1、循环的嵌套

C. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

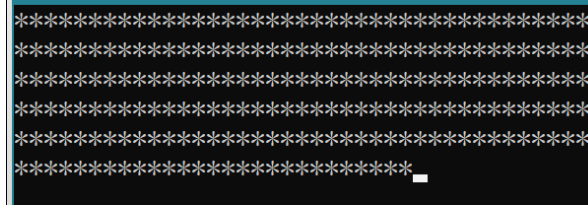
```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
    int i, j, count = 0;
    for(i=1; i<=100; i++) {
        for(j=1; i<=100; j++) {
            ++count;
            if (count % 1000 == 0) {
                cout << "*";
                _getch();
            }
        }
    }

    cout << "count = " << count << endl;
    return 0;
}
```

//注意：这个程序无法通过按CTRL+C终止，要关窗口

1、贴运行结果（能表现出要表达的意思即可）



每输入字符便会输出*

2、按内外for循环的执行步骤依次分析，为什么会得到这个结果？

第1步 - i=1, j=1, count=1

第2步 - i=1, j=2, count=2

...

第999步 - i=1, j=999, count=999

第1000步 - i=1, j=1000, count=1000

count%1000==0, 输出 * , 运行_getch(), 程序等待用户输入字符, 进行下一步操作

第1001步 - i=1, j=1000, count=1001

...

第2000步 - i=1, j=2000, count=2000

count%1000==0, 输出 * , 运行_getch(), 程序等待用户输入字符, 进行下一步操作

... ..

由于内层循环中i的值并未更改, i<=100永远为真, 所以内层循环一直执行, 每1000次暂停输出 * , 并等待用户输入。



§. 基础知识题 – 循环结构

此页不要删除，也没有意义，仅仅为了分隔题目



§. 基础知识题 – 循环结构

2、break与continue

A. 已知代码如下，回答问题

```
while(1) {  
    ①  
    ②  
    if (X)  
        continue;  
    ③  
    ④  
}
```

当X为真时，重复执行___ ①② ___ (①②③④)
当X为假时，重复执行___ ①②③④ ___ (①②③④)

```
for(1; 1; ④) {  
    ①  
    ②  
    if (X)  
        continue;  
    ③  
}
```

当X为真时，重复执行___ ①②④ ___ (①②③④)
当X为假时，重复执行___ ①②③④ ___ (①②③④)



§. 基础知识题 – 循环结构

2、break与continue

B. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <iostream>
using namespace std;

int main()
{
    int i=0, sum=0;

    while(i<1000) {
        i++;
        break;
        sum=sum+i;
    }

    cout << "i=" << i << endl;
    cout << " sum=" << sum << endl;

    return 0;
}
```

i=1
sum=0

//问题1: 循环执行了多少次? 1
//问题2: sum=sum+i执行了多少次? 0

```
#include <iostream>
using namespace std;

int main()
{
    int i=0, sum=0;

    while(i<1000) {
        i++;
        continue;
        sum=sum+i;
    }

    cout << "i=" << i << endl;
    cout << " sum=" << sum << endl;

    return 0;
}
```

i=1000
sum=0

//问题1: 循环执行了多少次? 1000
//问题2: sum=sum+i执行了多少次? 0

§. 基础知识题 – 循环结构



此页不要删除，也没有意义，仅仅为了分隔题目



§. 基础知识题 – 循环结构

3、观察程序运行结果

A. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)

```
#include <iostream>
#include <iomanip>    //格式输出
#include <cmath>      //fabs
#include <windows.h>  //取系统时间V
using namespace std;
```

```
int main()
{
```

```
    int s=1;
    double n=1, t=1, pi=0;
```

```
    LARGE_INTEGER tick, begin, end;
    QueryPerformanceFrequency(&tick);    //取计数器频率
    QueryPerformanceCounter(&begin);      //取初始硬件定时器计数
```

```
    while(fabs(t)>1e-6) {
        pi=pi+t;
        n=n+2;
        s=-s;
        t=s/n;
    }
```

```
    QueryPerformanceCounter(&end); //获得终止硬件定时器计数
```

```
    pi=pi*4;
    cout << "n=" << setprecision(10) << n << endl;
    cout<<"pi="<<setiosflags(ios::fixed)<<setprecision(9)<<pi<< endl;
```

```
    cout << "计数器频率: " << tick.QuadPart << "Hz" << endl;
    cout << "时钟计数 : " << end.QuadPart - begin.QuadPart << endl;
    cout << setprecision(6) << (end.QuadPart - begin.QuadPart)/double(tick.QuadPart) << "秒" <<endl;
```

```
    return 0;
}
```

用下面的迭代公式求Pi的值

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

(1) n, t, pi为double型

精度为1e-6: n=1000001 pi=3.141590654 时间=0.001861(秒)

1e-7: n=10000001 pi=3.141592454 时间=0.016094(秒)

1e-8: n=100000001 pi=3.141592634 时间=0.159503(秒)

1e-9: n=1000000001 pi=3.141592652 时间=1.757112(秒)

(因为机器配置不同，时间值可能不同)

(2) n, t, pi为float型

精度为1e-6: n=1000001 pi=3.141593933 时间=0.016459(秒)

1e-7: n=10000001 pi=3.141596556 时间=0.150481(秒)

1e-8: n= pi= 时间= (秒)

问: 1、7项中哪个没结果? 为什么?

float型，精度为1e-8时。float的精度仅限于6位有效数字，数据在达到小数点6位后便不可信，在精度为1e-8时数字不再发生变化，导致绝对值一直在循环合理范围内，无法跳出循环，无法显示结果。

2、float和double同进度下那个时间快? (观察现象即可，不需要解释原因)

double型

本页结果不要截图，手填即可



§. 基础知识题 – 循环结构

3、观察程序运行结果

B. 观察下列程序的运行结果，回答问题并将程序的运行结果截图贴上(如果有错则贴错误信息截图)，suoyi

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int n = 0, i, m, k;
    bool prime;
    for (m = 101; m <= 200; m += 2) { //偶数没必要判断
        prime = true;                //对每个数, 先认为是素数
        k = int(sqrt(m));
        for (i = 2; i <= k; i++)
            if (m % i == 0) {
                prime = false;
                break;
            }

        if (prime) {
            cout << setw(5) << m;
            n = n + 1;                //计数器, 只为了加输出换行
            if (n % 10 == 0)          //每10个数输出一行
                cout << endl;
        }
    } //end of for

    return 0;
}
```

(1) 目前输出结果: 一共21个, 每10个一行

Microsoft Visual Studio 调试控制台

```
101  103  107  109  113  127  131  137  139  149
151  157  163  167  173  179  181  191  193  197
199
```

(2) 将m的初值从101改为103, 应该是20个, 共2行
实际呢? 为什么?

```
103  107  109  113  127  131  137  139  149  151
157  163  167  173  179  181  191  193  197  199
```

实际为20个, 共4行。

m初值为101时, 在判断149是素数之后的151也是素数, 所以prime为真, 执行计数器操作, 执行换行操作, 但此时n=11, 并不会输出换行;

m初值为103时, 在判断151是素数之后的153不是素数, 所以prime为假, 不执行计数器操作, 但执行换行操作, 而此时n=10, 输出换行; 之后的155也不是素数, 同理此时n=10, 输出换行; 直到157是素数, n=11时才不输出换行, 于是多输出了两个换行。

(3) 将左侧程序改正确

(正确程序贴图在左侧, 覆盖现有内容即可)

§. 基础知识题 – 循环结构



此页不要删除，也没有意义，仅仅为了分隔题目