计算机学院 高级程序语言设计 课程实验报告

实验题目：C++开发环境与简单程序设计		学号：202300130150
日期：2324-3-12	班级： 4	姓名： 王成意

实验目的：

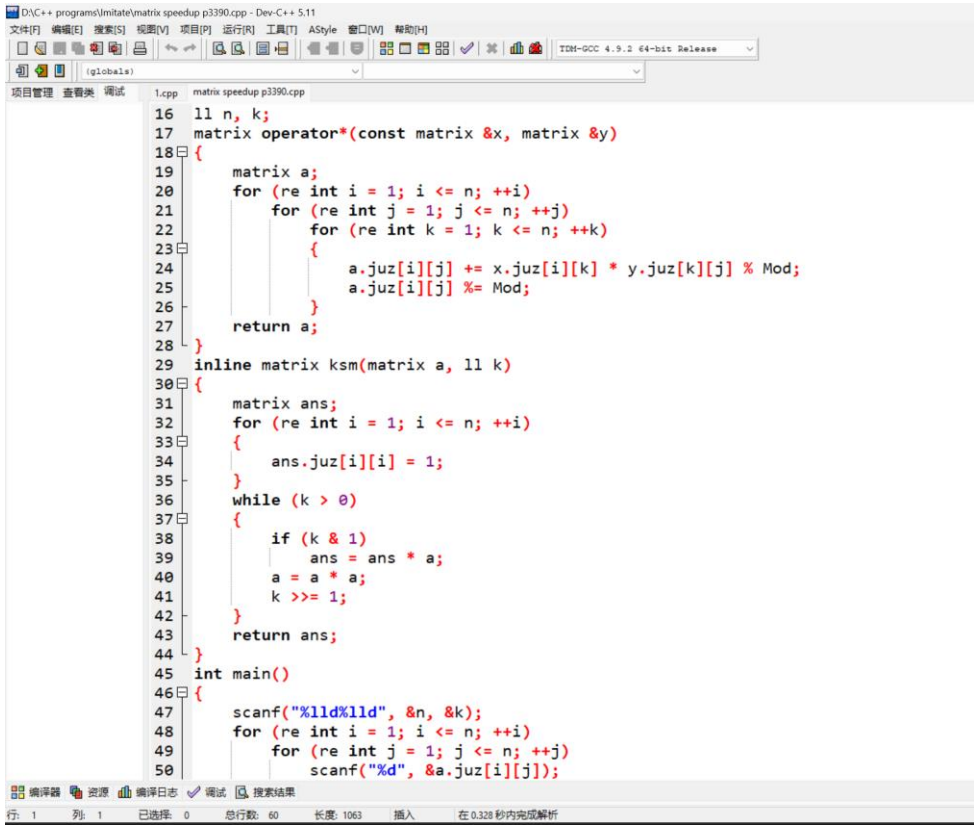
1. 熟悉 C++开发环境，能编写、调试程序，配置编译器。
2. 练习简单的 C++程序设计，熟悉 C++语法特点，掌握程序调试方法。

实验步骤与内容：

1. 练习在几种不同编译环境下，总结它们各自的特点和不同。

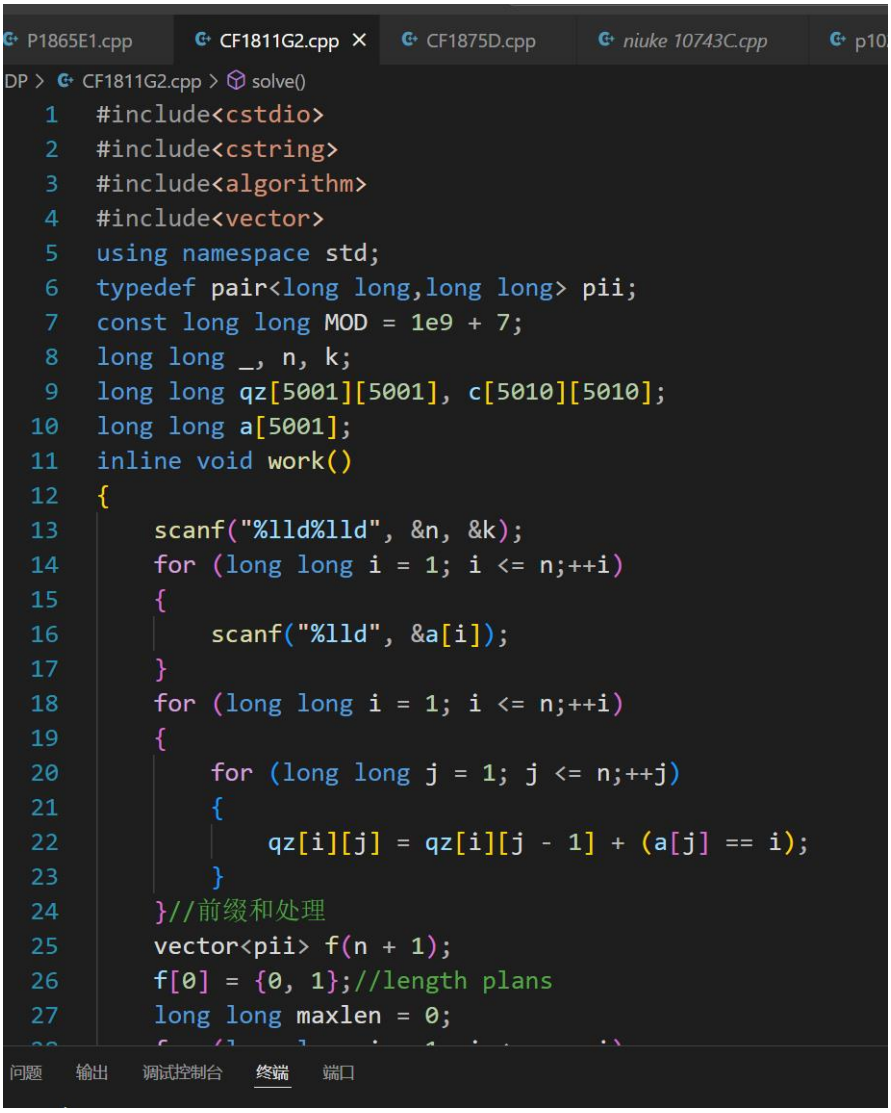
如：DEV-CPP, VS Code, VC，Clion 等 IDE 中的两种或以上，体验它们的不同。

DEV-C++

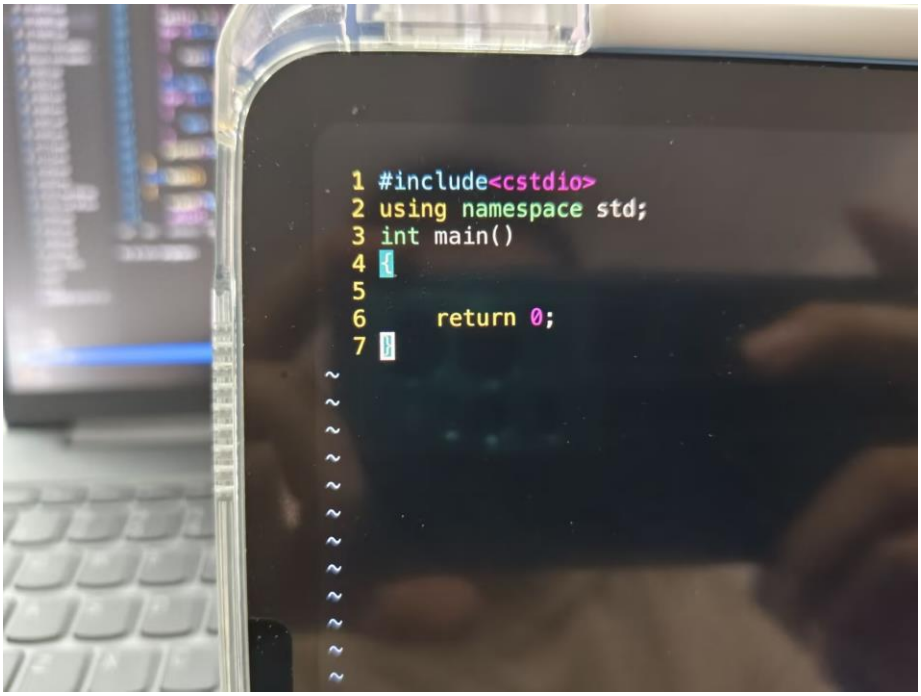


```
ll n, k;
matrix operator*(const matrix &x, matrix &y)
{
    matrix a;
    for (re int i = 1; i <= n; ++i)
        for (re int j = 1; j <= n; ++j)
            for (re int k = 1; k <= n; ++k)
            {
                a.juz[i][j] += x.juz[i][k] * y.juz[k][j] % Mod;
                a.juz[i][j] %= Mod;
            }
    return a;
}
inline matrix ksm(matrix a, ll k)
{
    matrix ans;
    for (re int i = 1; i <= n; ++i)
    {
        ans.juz[i][i] = 1;
    }
    while (k > 0)
    {
        if (k & 1)
            ans = ans * a;
        a = a * a;
        k >>= 1;
    }
    return ans;
}
int main()
{
    scanf("%lld%lld", &n, &k);
    for (re int i = 1; i <= n; ++i)
        for (re int j = 1; j <= n; ++j)
            scanf("%d", &a.juz[i][j]);
```

Vscode


P1865E1.cpp
CF1811G2.cpp
CF1875D.cpp
niuke 10743C.cpp
DP > CF1811G2.cpp > solve()
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
using namespace std;
typedef pair<long long,long long> pii;
const long long MOD = 1e9 + 7;
long long _, n, k;
long long qz[5001][5001], c[5010][5010];
long long a[5001];
inline void work()
{
scanf("%lld%lld", &n, &k);
for (long long i = 1; i <= n;++i)
{
scanf("%lld", &a[i]);
}
for (long long i = 1; i <= n;++i)
{
for (long long j = 1; j <= n;++j)
{
qz[i][j] = qz[i][j - 1] + (a[j] == i);
}
}//前缀和处理
vector<pii> f(n + 1);
f[0] = {0, 1};//length plans
long long maxlen = 0;
问题
输出
调试控制台
终端
端口

iPad：ISH

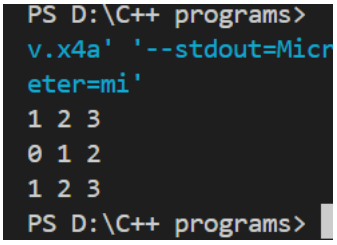


在 IDE 中运行下列两段代码，截屏运行结果并解释。

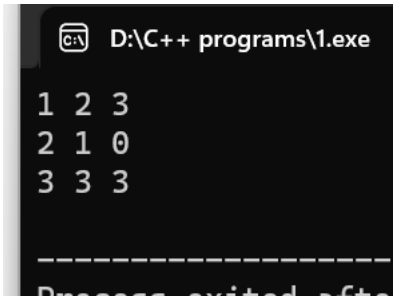
//代码 1

解释：
X 的编码为 0xffffffff，在 int 里第一位 1 是符号位，被当成了负号，剩下的数组成 2^31-1；y=x，而 y 是 unsigned int，第一位不是符号位，表示为 2^32-1;z 编码为 0x7fffffff，+1 后是 0x10000000，在 int 中是-2^31.

//代码 2
Vscode：



DEV-C++：



对于第一行和第三行很好理解。
第一行只能是 1 2 3；
第三行两个编译器都是先执行++操作，最后执行 cout 操作；
而对于第二行，两种编译器做出了不同的应对：
Vscode 反汇编：

```
6               cout<<b++<<" "<<b++<<" "<<b++<<endl;
=> 0x00000000004015ab <+91>:    mov    -0x8(%rbp),%eax
   0x00000000004015ae <+94>:    lea    0x1(%rax),%edx
   0x00000000004015b1 <+97>:    mov    %edx,-0x8(%rbp)
   0x00000000004015b4 <+100>:   mov    %eax,%edx
   0x00000000004015b6 <+102>:   mov    0x2d53(%rip),%rcx        # 0x404310 <__fu0__ZSt4cout>
   0x00000000004015bd <+109>:   callq  0x401720 <_ZNSolsEi>
   0x00000000004015c2 <+114>:   lea    0x2a41(%rip),%rdx        # 0x40400a <_ZStL19piecewise_con
   0x00000000004015c9 <+121>:   mov    %rax,%rcx
   0x00000000004015cc <+124>:   callq  0x401700 <_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT
   0x00000000004015d1 <+129>:   mov    %rax,%rcx
   0x00000000004015d4 <+132>:   mov    -0x8(%rbp),%eax
   0x00000000004015d7 <+135>:   lea    0x1(%rax),%edx
   0x00000000004015da <+138>:   mov    %edx,-0x8(%rbp)
   0x00000000004015dd <+141>:   mov    %eax,%edx
   0x00000000004015df <+143>:   callq  0x401720 <_ZNSolsEi>
   0x00000000004015e4 <+148>:   lea    0x2a1f(%rip),%rdx        # 0x40400a <_ZStL19piecewise_con
   0x00000000004015eb <+155>:   mov    %rax,%rcx
   0x00000000004015ee <+158>:   callq  0x401700 <_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT
   0x00000000004015f3 <+163>:   mov    %rax,%rcx
   0x00000000004015f6 <+166>:   mov    -0x8(%rbp),%eax
   0x00000000004015f9 <+169>:   lea    0x1(%rax),%edx
   0x00000000004015fc <+172>:   mov    %edx,-0x8(%rbp)
   0x00000000004015ff <+175>:   mov    %eax,%edx
   0x0000000000401601 <+177>:   callq  0x401720 <_ZNSolsEi>
   0x0000000000401606 <+182>:   mov    0x2d13(%rip),%rdx        # 0x404320 <.refptr._ZSt4endlIcS
   0x000000000040160d <+189>:   mov    %rax,%rcx
   0x0000000000401610 <+192>:   callq  0x401728 <_ZNSolsEPFRSoS_E>
```

可见，vscode 每执行一个 cout 之后就会立刻执行++操作；

```
=> 0x000000000040158d <+93>:    mov    -0x8(%rbp),%ebx
   0x0000000000401590 <+96>:    lea    0x1(%rbx),%eax
   0x0000000000401593 <+99>:    mov    %eax,-0x8(%rbp)
   0x0000000000401596 <+102>:   mov    -0x8(%rbp),%esi
   0x0000000000401599 <+105>:   lea    0x1(%rsi),%eax
   0x000000000040159c <+108>:   mov    %eax,-0x8(%rbp)
   0x000000000040159f <+111>:   mov    -0x8(%rbp),%eax
   0x00000000004015a2 <+114>:   lea    0x1(%rax),%edx
   0x00000000004015a5 <+117>:   mov    %edx,-0x8(%rbp)
   0x00000000004015a8 <+120>:   mov    %eax,%edx
   0x00000000004015aa <+122>:   mov    0x8b0df(%rip),%rcx        # 0x48c690 <.refptr._ZSt4cout>
   0x00000000004015b1 <+129>:   callq  0x44d5f0 <_ZNSolsEi>
   0x00000000004015b6 <+134>:   lea    0x86a4c(%rip),%rdx        # 0x488009
   0x00000000004015bd <+141>:   mov    %rax,%rcx
   0x00000000004015c0 <+144>:   callq  0x46ed00 <_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PK
   0x00000000004015c5 <+149>:   mov    %esi,%edx
   0x00000000004015c7 <+151>:   mov    %rax,%rcx
   0x00000000004015ca <+154>:   callq  0x44d5f0 <_ZNSolsEi>
   0x00000000004015cf <+159>:   lea    0x86a33(%rip),%rdx        # 0x488009
   0x00000000004015d6 <+166>:   mov    %rax,%rcx
   0x00000000004015d9 <+169>:   callq  0x46ed00 <_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PK
   0x00000000004015de <+174>:   mov    %ebx,%edx
   0x00000000004015e0 <+176>:   mov    %rax,%rcx
   0x00000000004015e3 <+179>:   callq  0x44d5f0 <_ZNSolsEi>
   0x00000000004015e8 <+184>:   mov    0x8b0b1(%rip),%rdx        # 0x48c6a0 <.refptr._ZSt4endlIcSt11cha
   0x00000000004015ef <+191>:   mov    %rax,%rcx
   0x00000000004015f2 <+194>:   callq  0x44d3f0 <_ZNSolsEPFRSoS_E>
   0x00000000004015f7 <+199>:   addl   $0x1,-0x4(%rbp)
   0x00000000004015fb <+203>:   addl   $0x1,-0x4(%rbp)
```

可见，dev-c++先将所有的 cout 操作压栈，再对 b++，然后将不同的 cout 按照栈中的顺序给出 b 的值，因此会出现 2 1 0.

2. 了解不同的编译器

自行阅读。

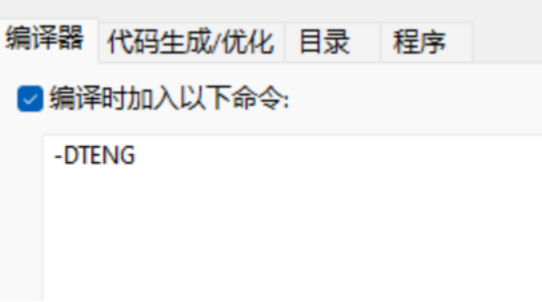
3. 学会配置 Dev C++编译选项：

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

int main() {
#ifdef TENG
    cout<<"teng"<<endl;
#else
    cout<<"no teng"<<endl;
#endif
    return 0;
}
```

```
D:\C++ programs\1.exe
no teng

--------------------------------
Process exited after 0.07357
请按任意键继续. . .
```




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

int main() {
#ifdef TENG
    cout<<"teng"<<endl;
#else
    cout<<"no teng"<<endl;
#endif
    return 0;
}
```

```
D:\C++ programs\1.exe
teng

--------------------------------
Process exited after 0.073
请按任意键继续. . .
```

为什么改变编译选项也可以改变程序的运行结果？

因为程序中有关于编译选项的判断：#ifdef TENG

结论分析与体会：

C++是一种非常巧妙与伟大的语言。不同的编译器可能会有不同的结果，同一个编译器加上不同的编译选项也会有不同的结果，我对后续的 c++学习充满了好奇。