|  |  |
| --- | --- |
| **编号:** | **第2章第1次** |

****

信息科学与工程学院实验报告

《面向对象程序设计》

**Object-Oriented Programming**

|  |  |
| --- | --- |
| 姓名： | 朱会琛 |
| 学号： | 202311000202 |
| 班级： | 计工本2302 |
| 教师: | 张庆科 |
| 时间： | 2024年10月10日 |

**《面向对象程序设计》实验报告**

**报告要求：**实验报告包含实验目的、实验内容、实验过程（详细操作流程）、实验结果（程序运行结果高清截图）、实验分析总结五个部分。报告中若涉及代码程序，请在附录部分提供完整程序源码及源码托管地址(基于Highlight软件导入源码)。报告撰写完毕后请将PDF格式版本上传到坚果云作业提交系统。

1. **实验目的**

\* 掌握使用C++面向过程程序设计方法

\* 掌握C++对C语言的典型改进和扩展

\* 掌握Visual Studio建立项目步骤

\* 掌握Visual Studio程序调试方法

\* 掌握Git的基本操作原理和常见命令

1. **实验内容**

**（1）任务一：论述题**  
阐述git的基本工作过程和工作原理，基于代码托管网站gitee或GitHub给出git仓库项目的创建过程。然后结合课程PPT有关git的内容，给出git常见的操作命令的使用方法。

**（2）任务二:  程序设计题- C++素数判断**

  建立VS项目, 在源文件```main.cpp```中定义素数判别函数```isPrimeNumber()```, 在主函数中输入一个整数m, 然后输出该整数是否为素数的信息.

测试案例:

--------------------------------  
 请输入任意一个整数: 17

 整数17是素数: True  
--------------------------------

**（3）任务三: 程序设计题- C++验证哥德巴赫猜想**

**哥德巴赫猜想**: 对于任意一个不小于6的偶数,均可以将其表示为两个素数之和,例如: 6=3+3, 8=3+5, 12=5+7,...等等. 请在主函数中输入一个不小于6的整数N, 然后调用偶数解码函数DecodeEvenToPrime(int num),输出小于整数N的所有偶数的素数分解形式.

程序运行测试案例:

---------------------------------  
请输入一个整数N (N>6): 20  
偶数20的素数分解结果: 20 = 3 + 17  
偶数20的素数分解结果: 20 = 7 + 13  
偶数18的素数分解结果: 18 = 5 + 13  
偶数18的素数分解结果: 18 = 7 + 11  
偶数16的素数分解结果: 16 = 3 + 13  
偶数16的素数分解结果: 16 = 5 + 11  
偶数14的素数分解结果: 14 = 3 + 11  
偶数14的素数分解结果: 14 = 7 + 7  
偶数12的素数分解结果: 12 = 5 + 7  
偶数10的素数分解结果: 10 = 3 + 7  
偶数10的素数分解结果: 10 = 5 + 5  
偶数8的素数分解结果: 8 = 3 + 5  
偶数6的素数分解结果: 6 = 3 + 3  
---------------------------------  
输出完毕!

\* 如果未按照要求输入,例如输入2,4,或者输入字符a, '\*' 等,程序会出现什么情况? 请继续修改完善解码函数,确保程序输入输出的健壮性.

**（4）任务四: 程序设计题- C++基本函数设计（选做题）**

使用C++语言设计一个随机数自动生成函数GenerateRandNumbers(...), 该函数可以在给定的数值区间范围内按照指定精度返回n个随机小数. 然后,基于上述结果,统计分析这n个随机数的均值和方差, 最后基于均值和方差结果分析随机数分布特点(开放性问题)

程序运行测试案例:

-----------随机数生成器---------  
\* 请输入数值区间a和b: 0 10  
\* 请输入数值精度位数: 2  
\* 请输入随机数的数目: 100  
-------------------------------  
随机数如下(每行输出5个数值):  
0.25 7.83 0.05 1.95 9.26  
5.84 5.42 2.99 10.00 8.10  
...

-------------------------------  
均值: xxx  
方差: xxx  
-------------------------------

输出完毕!

1. **实验过程**

**实验一：论述题**  
阐述git的基本工作过程和工作原理，基于代码托管网站gitee或GitHub给出git仓库项目的创建过程。然后结合课程PPT有关git的内容，给出常见git常见的操作命令的使用方法。

答：git 的基本工作过程和原理

1.工作过程

首先是工作区（Workspace），开发者在这里进行代码的编辑修改。例如在本地电脑上打开代码文件进行编写新功能或修复 Bug 等操作。

然后通过git add命令将修改的文件添加到暂存区（Staging Area/Index）。暂存区可以看作是一个准备提交的区域，它收集了要提交的文件修改内容。

接着使用git commit命令将暂存区的内容提交到本地仓库（Local Repository），本地仓库会记录每次提交的版本信息，包括提交的时间、作者、提交说明以及代码的改动情况等。

最后通过git push命令将本地仓库的内容推送到远程仓库（Remote Repository，如 GitHub 上的仓库），这样其他开发者就可以获取到最新的代码。

2.工作原理

git 是分布式版本控制系统，每个开发者的本地电脑上都有一个完整的仓库副本。这意味着即使没有网络连接，开发者仍然可以在本地进行版本控制操作，如提交、查看历史版本等。

它使用一种基于哈希算法的对象存储方式。每个文件的每次修改都会生成一个新的对象，这些对象通过哈希值进行唯一标识，并存储在.git目录下的objects文件夹中。同时，git通过维护一个索引文件（.git/index）来记录文件的状态和指向对应的对象。

基于 GitHub 的 git 仓库项目创建过程

注册 GitHub 账号，如果是学生可以使用校园邮箱注册。

登录 GitHub 后，点击右上角的 “+” 号，选择 “New repository”。

在创建仓库页面，填写仓库名称（如 “OOP\_Homework”），可以添加一些描述信息，选择仓库的公开或私有属性等，然后点击 “Create repository” 按钮。

常见 git 操作命令使用方法

3.配置信息

初次使用 git 前需要配置用户信息，包括用户名和邮箱。例如：

git config --global user.name "tsingke"设置用户名。

git config --global user.email "tsingkesdnu.edu.cn"设置邮箱。

4.仓库操作

创建本地仓库：

可以使用git init命令在本地初始化一个空的 git 仓库。例如在一个项目文件夹下执行该命令，就会在该文件夹下创建一个.git隐藏目录，表示该文件夹已经是一个 git 仓库。

也可以使用git clone命令从远程仓库克隆一个项目到本地。例如git clone https://github.com/username/repository.git，其中https://github.com/username/repository.git是远程仓库的地址。

查看仓库状态：使用git status命令可以查看当前仓库中文件的状态，如哪些文件被修改了、哪些文件是新增的、哪些文件已经添加到暂存区等。

5.文件操作

添加文件到暂存区：使用git add命令。例如git add.可以将当前目录下所有修改的文件添加到暂存区。如果只想添加某个特定文件，可以使用git add filename。

提交文件到本地仓库：使用git commit命令，同时需要添加提交说明。例如git commit -m "修复了xx Bug"，这里的-m参数后面跟着提交说明。

删除文件：如果要删除文件，首先在工作区删除文件（如rm filename），然后使用git add更新暂存区（可以使用git add.或者git add filename，如果是删除单个文件），最后使用git commit提交删除操作到本地仓库（如git commit -m "delete filename"）。也可以使用git rm filename命令直接从工作区和暂存区同时删除文件，然后再提交删除操作到本地仓库。

版本回退

6.git reset命令用于版本回退，有以下几种方式：

git reset --soft：软回退，会保留工作区和暂存区的修改，只是将当前分支的 HEAD 指针回退到指定版本。例如git reset --soft HEAD~1会回退到上一个版本。

git reset --hard：硬回退，会丢弃工作区和暂存区的修改，将工作区、暂存区和本地仓库都回退到指定版本。例如git reset --hard HEAD~2会回退到上上个版本。

git reset --mixed：混合回退，默认参数，会保留工作区的修改，将暂存区回退到指定版本，同时将当前分支的 HEAD 指针回退到指定版本。例如git reset --mixed HEAD~3会回退到上上上个版本。

7.分支操作

创建分支：使用git branch <分支名称>命令创建一个新的分支。例如git branch feature-login-page创建一个名为 “feature-login-page” 的功能分支。

查看分支：使用git branch命令查看当前仓库中所有的分支。

切换分支：使用git switch <分支名称>命令切换到指定分支。例如git switch feature-login-page切换到 “feature-login-page” 分支。

合并分支：使用git merge <分支名称>命令将指定分支合并到当前分支。例如将 “feature-login-page” 分支合并到 “master” 分支，可以使用git merge feature-login-page。

删除分支：如果分支已经合并过，可以使用git branch -d <分支名称>命令删除。例如git branch -d feature-login-page。如果分支未合并过，需要使用git branch -D <分支名称>命令删除。

8.查看差异

git diff命令用于查看文件差异：

默认情况下git diff查看工作区和暂存区之间的差异。

git diff HEAD查看工作区和本地仓库之间的差异。

通过git diff命令还可以查看不同分支之间的差异（如git diff branch1 branch2）以及不同版本之间的差异（如git diff version1 version2，可以使用版本号或者HEAD的别名来指定版本）。

**实验二： 程序设计题- C++素数判断**

建立VS项目, 在源文件```main.cpp```中定义素数判别函数```isPrimeNumber()```, 在主函数中输入一个整数m, 然后输出该整数是否为素数的信息.

测试案例:

--------------------------------  
  请输入任意一个整数: 17

 整数17是素数: True

**源码：main.c**

#include <iostream>

using namespace std;

bool isPrimeNumber(int x){

if (x <= 1)

return false;

for (int i = 2; i \* i <= x;i++){

if (x % i == 0)

return false;

}

return true;

}

int main(){

int m;

cout << "请输入任意一个整数:";

cin >> m;

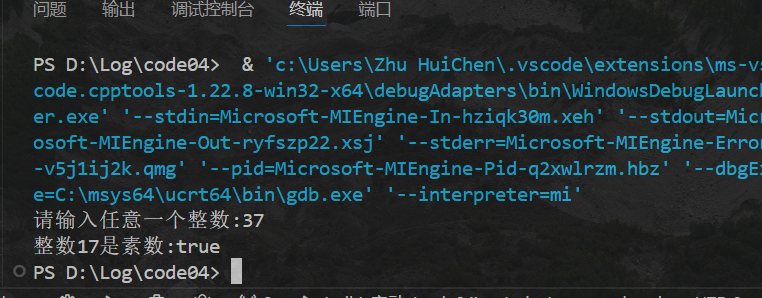
// 使用流操纵器来控制输出的是false和true

cout << boolalpha <<"整数17是素数:"<< isPrimeNumber(m) << endl;

return 0;

}

**运行结果：**



**实验三：程序设计题- C++验证哥德巴赫猜想**

**哥德巴赫猜想**: 对于任意一个不小于6的偶数,均可以将其表示为两个素数之和,例如: 6=3+3, 8=3+5, 12=5+7,...等等. 请在主函数中输入一个不小于6的整数N, 然后调用偶数解码函数DecodeEvenToPrime(int num),输出小于整数N的所有偶数的素数分解形式.

程序运行测试案例:

---------------------------------  
请输入一个整数N (N>6): 20  
偶数20的素数分解结果: 20 = 3 + 17  
偶数20的素数分解结果: 20 = 7 + 13  
偶数18的素数分解结果: 18 = 5 + 13  
偶数18的素数分解结果: 18 = 7 + 11  
偶数16的素数分解结果: 16 = 3 + 13  
偶数16的素数分解结果: 16 = 5 + 11  
偶数14的素数分解结果: 14 = 3 + 11  
偶数14的素数分解结果: 14 = 7 + 7  
偶数12的素数分解结果: 12 = 5 + 7  
偶数10的素数分解结果: 10 = 3 + 7  
偶数10的素数分解结果: 10 = 5 + 5  
偶数8的素数分解结果: 8 = 3 + 5  
偶数6的素数分解结果: 6 = 3 + 3  
---------------------------------

源码：#include <iostream>

using namespace std;

int N;

bool isEven(int ans){

return (ans % 2 == 0);

}

bool isPrimeNumber(int x){

if (x <= 1)

return false;

for (int i = 2; i \* i <= x;i++){

if (x % i == 0)

return false;

}

return true;

}

void DecodeEvenToPrime(int num){

for (int j = 2; j <= num / 2; j++){

if (isPrimeNumber(j) && isPrimeNumber(num - j)){

cout << "偶数" << num << "的素数分解结果为:" << num << "=" << j << '+' << num - j << endl;

}

}

}

int main(){

cout << "请输入一个整数N(N > 6):";

cin >> N;

if (isEven(N)){

for (int k = N; k >= 6;k -= 2){

DecodeEvenToPrime(k);

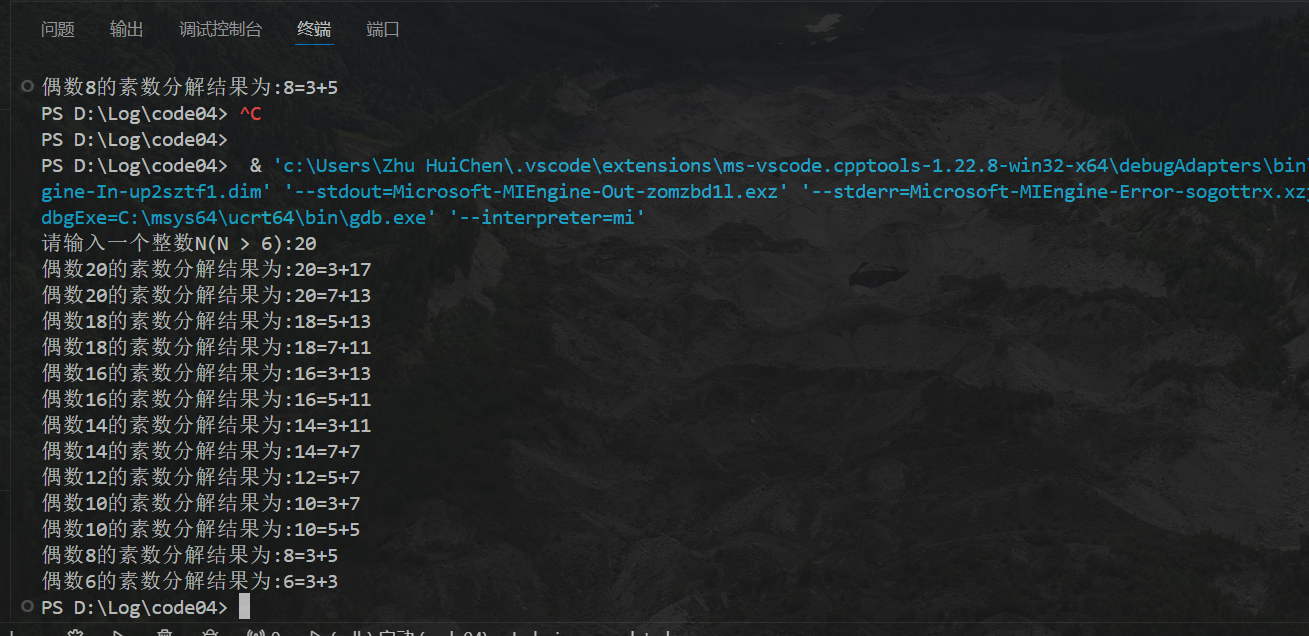
}

}

return 0;

}

**运行结果：**

****

**实验四**：**程序设计题- C++基本函数设计（选做题）**

使用C++语言设计一个随机数自动生成函数GenerateRandNumbers(...), 该函数可以在给定的数值区间范围内按照指定精度返回n个随机小数. 然后,基于上述结果,统计分析这n个随机数的均值和方差, 最后基于均值和方差结果分析随机数分布特点(开放性问题)

程序运行测试案例:

-----------随机数生成器---------  
\* 请输入数值区间a和b: 0 10  
\* 请输入数值精度位数: 2  
\* 请输入随机数的数目: 100  
-------------------------------  
随机数如下(每行输出5个数值):  
0.25 7.83 0.05 1.95 9.26  
5.84 5.42 2.99 10.00 8.10  
...

-------------------------------  
均值: xxx  
方差: xxx  
-------------------------------

输出完毕!

源码：

#include <iostream>

#include <random>

#include <ctime>

#include <iomanip>

#include <cmath>

using namespace std;

void GenerateRandNumbers(int l, int r, int acc, int cnt, double &mean, double &variance) {

random\_device rd;

mt19937 gen(rd());

// 控制范围

uniform\_real\_distribution<double> dis(l, r);

// 生成随机数并计算均值和方差

double sum = 0.0;

double sum\_sq\_diff = 0.0;

mean = 0.0;

variance = 0.0;

int ans = 0;

for (int i = 0; i < cnt; ++i) {

double random\_number = dis(gen);

sum += random\_number;

sum\_sq\_diff += (random\_number \* random\_number);

// 控制小数位并输出

cout << fixed << setprecision(acc) << random\_number << ' ';

ans++;

if (ans % 5 == 0)

puts("");

}

// 计算均值

mean = sum / cnt;

// 计算方差

variance = (sum\_sq\_diff / cnt) - (mean \* mean);

}

int main() {

int a, b, c, num;

double mean, variance;

cout << "--------随机数生成器--------" << endl;

cout << "\*请输入数值区间a和b:";

cin >> a >> b;

cout << "\*请输入数值精度位数:";

cin >> c;

cout << "\*请输入随机数的数目:";

cin >> num;

cout << "随机数如下:" << endl;

GenerateRandNumbers(a, b, c, num, mean, variance);

cout << "----------------------" << endl;

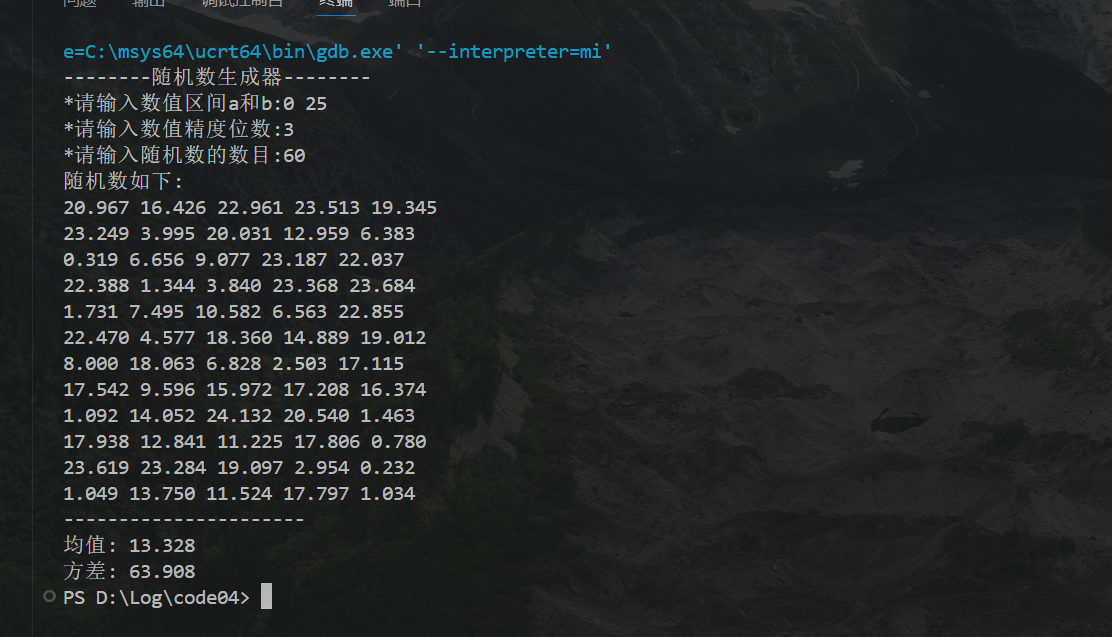
cout << "均值: " << fixed << setprecision(c) << mean << endl;

cout << "方差: " << fixed << setprecision(c) << variance << endl;

return 0;

}

**运行结果：**

****

1. **实验总结**

收获很大，我觉得很有意思！既提高了代码的熟练度又运用代码解决了有趣的数学问题，在任务三的实验过程中，我第一遍执行程序的时候，误将传递的参数写为N，导致整个程序陷入循环，重复输出20 = 3 + 17 和 20 = 7 + 13,于是我在N 和 k 处打断点，开始debug，最终找到错误的点并且及时的更正，通过最后的选做题，我了解到生成随机数的方法，精准的控制小数位数。在实验结束后，我准备将这次的源码全部传到我的GitHub仓库中！