**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软件工程2105

学 号： 8202210810

姓 名： 阎庆阳

**实验报告成绩：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验** | **实验一** | **实验二** | **实验三** | **实验四** | **实验五** | **总评** |
| **成绩** |  |  |  |  |  |  |

**批阅教师：**

**实验一、实验环境与简单程序设计**

**一、实验目的**

1、掌握集成开发环境，掌握C++程序的基本要素以及完整的C++程序开发过程。

2、掌握基本数据类型、运算符和表达式的使用。理解隐式转换和强制转换，理解数据超过该数据类型

表示范围时的溢出。掌握不同数据之间的混合算术运算中数据类型的转换。

3、变量的定义与常量的使用。

4、输入、输出的实现。

5、编译信息的理解与错误的修改。

6、简单程序的设计。

**二、实验内容**

熟悉C++编程环境，可以使用VS；对已经能熟练掌握C++开发环境的同学，可

以跳过本部分内容）

**1.编辑输入下列程序，找出下面代码的错误并改正：**

#include<iostream>

using namespace std;

int main()

{

Int i = k + 1;

cout << i++ << endl;

int i = 1;

cout << i++ << endl;

cout << "Welcome to C++"<<endl;

return 0;

}

**2.求圆锥的体积：要求键盘输入圆锥底的半径、锥高，使用标识符常量定义圆周率。**

(1)创建一个控制台项目

(2)在文件中输入程序内容，存盘

(3)编译、连接、运行；观察结果

**3**.**通过下面程序验证你所使用系统上运行的C++编译器中每个基本数据类型的长度。**

#include<iostream>

using namespace std;

int main()

{

cout << "char length:" << sizeof(char) << endl;

cout << "int length:" << sizeof(int) << endl;

}

**4.观察下面程序的执行结果。**

#include <iostream>

#include <iomanip>

using namespace std;

int main()

unsigned int testUnint=65534;//oxfffe

cout << "output in unsigned int 1 type:"" << testUnint<< end;//<<oct;

cout << "output in char type:!" << static\_ cast<char>(testUnint)<< endl;

cout << "output in short type:" << static\_ cast<short>(testUnint)<< endl;//为什么结果为-2?:

cout << "output in int type:" << static\_ cast<int>. (testUnint)<< endl;

cout << "output in double type:"<< static cast<double>(testUnint)<< endl;

cout << "output in double type:" <<setprecision(4)<< static\_ cast<double>(testUnint)<< endl;

cout << "output in Hex unsigned int type:" <<hex<< testUnint<< endl; //16进制输出

system("pause");

return 0;

**自己编程测试一下将testUnint按8进制输出<<oct;je\_将一个实数转换成int,观察结果。**

**5.编程，输入华氏温度，将其转换为摄氏温度后输出（保留两位小数）。**

**三、算法分析，程序结果**

#### 1.

错误1：Int应改为int

错误2：Int i = k + 1处，k未提前声明。

错误3：第二处int i = 1中，i重复声明

#### 2.

**代码：**

#include<iostream>

using namespace std;

const double PI = 3.14159; // 定义圆周率为常量

int main() {

double radius, height;

cout << "请输入圆锥底的半径：";

cin >> radius;

cout << "请输入圆锥的高：";

cin >> height;

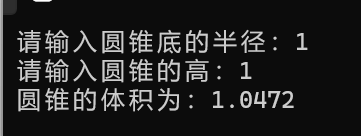
double volume = (PI \* radius \* radius \* height) / 3.0; // 计算圆锥体积

cout << "圆锥的体积为：" << volume << endl;

return 0;

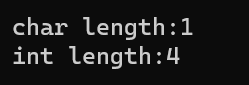
}

**运行结果截图：**



#### 3.

运行结果：



#### 4.

代码：

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

unsigned int testUnint = 65534;//oxfffe

cout << "output in unsigned int 1 type:" << testUnint << endl;//<<oct;

cout << "output in char type:!" << static\_cast <char>(testUnint) << endl;

cout << "output in short type:" << static\_cast<short>(testUnint) << endl;//为什么结果为-2?:

cout << "output in int type:" << static\_cast<int>(testUnint) << endl;

cout << "output in double type:" << static\_cast<double>(testUnint) << endl;

cout << "output in double type:" << setprecision(4) << static\_cast<double>(testUnint) << endl;

cout << "output in Hex unsigned int type:" << hex << testUnint << endl; //16进制输出

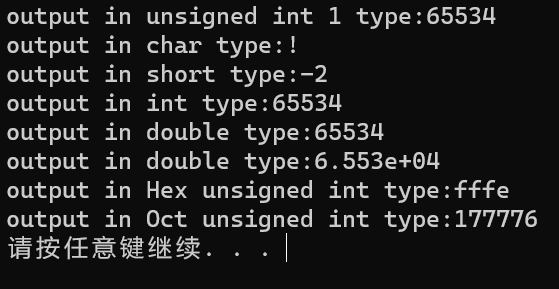
cout << "output in Oct unsigned int type:" << oct << testUnint << endl; //8进制输出

system("pause");

return 0;

}

运行结果：其中最后一行输出为8进制输出



static\_cast<short>(testUnint)的结果为2的原因： short类型是有符号整数类型，它在大多数平台上是16位的，范围为-32768~32767。由于在进行转换时，值65534（0xfffe）的最高位为1，所以被认为是补码。

#### 5.

代码：

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

double fahrenheit, celsius;

while (true) {

char c;

cout << "请输入华氏温度: ";

cin >> fahrenheit;

celsius = (fahrenheit - 32) \* 5 / 9;

cout << "摄氏温度为: " << fixed << setprecision(2) << celsius << endl;

cout << endl;

cout << "是否继续[y/n]：";

cin >> c;

cout << endl;

if (c == 'n')

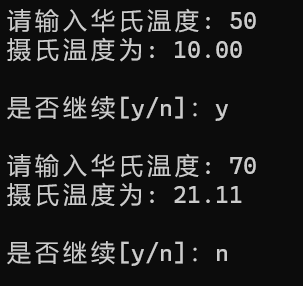
break;

}

return 0;

}

运行结果：



**四、遇到的问题与解决方法**

在运行第四题的代码时，由于之前未使用过static\_cast方法进行类型转换，尝试利用atoi、itoa等方法进行类型转换，但发现static\_cast方法可以进行多种类型转换，极大提高了代码的简洁度与可理解性。

**五、体会**

这次的总体上相对容易实现，主要涉及了C++的一些基本的语法知识，如第一题的语法纠正，第二题与第五题的编程。当然收获最大的还是第四题中的各个类型转换，因为在日常编程中很少能够涉及到进制之间的转换，所以还是第一次接触到static\_cast <char>这样的类型转换！“纸上得来终觉浅，绝知此事要躬行”，我相信在一次次实验的基础上，C++的编程能力会一直得到提升！

**实验二、数据结构**

**一、实验目的**

1、学习与掌握逻辑运算与逻辑表达式。

2、熟练掌握if、switch、while、do-while，for语句的语法结构与执行过程。

3、掌握选择、循环程序的设计方法

**二、实验内容**

1、输入一个字符，如果为小写，转换为大写输出，否则，输出其后继字符的ASCII码值。

2、输入x计算表达式的值：



分别输入 0.2, 1, 5 , 0,观察输出结果。

1. 输入三角形的三条边，求周长，并判断该三角形是否为等腰三角形（提示：要三边是否可以构成三角形）。

4、 完成计算器程序，实现（+ - \* / %）运算。考虑除数为 0 与运算符非法的情况。

5、输入一行字符，分别统计出其中英文字母、空格、数字字符和其它字符的个数。

提示：从键盘上读入一个字符给变量 c，判断 c 是属于哪种字符并计数，循环读入下个字符，直到回车换行字符'\n'为止。

cin，scanf（）都不能读入空格以及‘\n’字符，查找资料解决输入这两个字符的方法。

（这个题训练大家自主学习能力以及如何获取新知识、探索解决未知问题的能力。）

6、编写一个程序：从键盘上输入两个正整数，求 a 和 b 的最大公约数与最小公倍数。

7、使用循环结构输出下列图形：

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

8、从键盘输入 a，用迭代法求 a 的平方根 x= *a* 。求平方根的迭代公式为：



要求精确到|xn+1 - xn|<10 -5。

提示：迭代法是把 xn代入迭代公式右边，计算出 xn+1来，然后把 xn+1 作为新的 xn ，计算出新的 xn+1，如此重复，直到|xn+1 - xn|<10 -5 时，xn+1 为所求的平方根。可以把 a 作为 xn 的初始值。

思考：（1）如果输入 a 为负，在运行时会出现什么情况? 修改程序使之能处理任何的 a 值。

(2）能否|xn+1 -xn|<10 -10或更小? 为什么? 请试一下。

9、苹果每个 0.8 元，第一天买 2 个，第二天开始，每天买前天的 2 倍，直到购买的苹果数不超过100的最大值，求每天平均花多少钱。

**三、算法分析，程序结果**

#### 1.

代码：

#include <iostream>

using namespace std;

int main() {

char ch;

cout << "请输入一个字符：";

cin >> ch;

if (ch >= 'a' && ch <= 'z') {

// 如果是小写字母，则转换为大写输出

char upper = ch - 32;

cout << "转换为大写字母：" << upper << endl;

}

else {

// 如果不是小写字母，则输出后继字符的ASCII码值

int ascii = static\_cast<int>(ch) + 1;

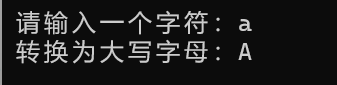
cout << "后继字符的ASCII码值：" << ascii << endl;

}

return 0;

}

运行结果：





#### 2.

代码：

#include <iostream>

using namespace std;

float f(float x) {

if (0 < x && x < 1) {

return 3 - 2 \* x;

}

else if (1 <= x && x < 5) {

return 0.5 / x + 1;

}

else if (5 <= x && x < 10) {

return x \* x;

}

else {

return NULL;

}

}

int main() {

float x;

cout << "请输出入一个大于0小于10的数：";

cin >> x;

while (true) {

float y = f(x);

if (y == NULL) {

cout << "请重新输入一个大于0小于10的数：";

cin >> x;

}

else {

cout << "y = " << y << endl;

break;

}

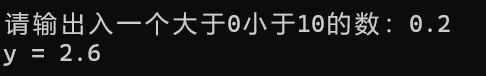
}

return 0;

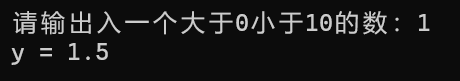
}

输出结果：

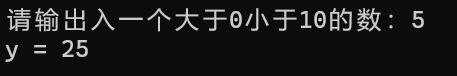
X = 0.2



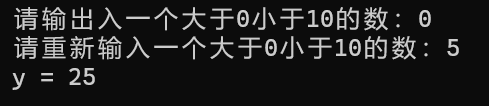
X = 1



X = 5



X = 0



#### 3.

代码：

#include <iostream>

using namespace std;

bool isTriangle(int a, int b, int c) {

// 判断三边是否可以构成三角形

if (a + b > c && a + c > b && b + c > a)

return true;

return false;

}

bool isIsoscelesTriangle(int a, int b, int c) {

// 判断是否为等腰三角形

if (a == b || a == c || b == c)

return true;

return false;

}

int main() {

int a, b, c;

cout << "请输入三角形的三条边：" << endl;

cout << "a = ";

cin >> a;

cout << "b = ";

cin >> b;

cout << "c = ";

cin >> c;

if (isTriangle(a, b, c)) {

int perimeter = a + b + c; // 计算周长

cout << "三角形的周长为：" << perimeter << endl;

if (isIsoscelesTriangle(a, b, c)) {

cout << "这是一个等腰三角形！" << endl;

}

else {

cout << "这不是一个等腰三角形！" << endl;

}

}

else {

cout << "这三条边不能构成三角形！" << endl;

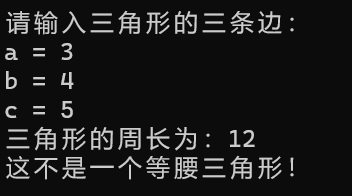
}

return 0;

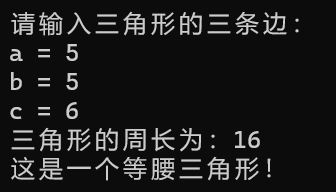
}

运行结果：

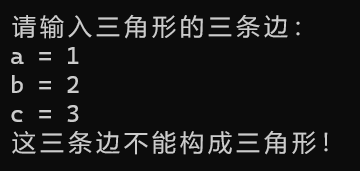
非等腰三角形



等腰三角形



非三角形



#### 4.

简易计算器代码：

#include <iostream>

using namespace std;

int main() {

char op;

double num1, num2, result;

cout << "请输入运算式: ";

cin >> num1 >> op >> num2;

switch (op) {

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

case '\*':

result = num1 \* num2;

break;

case '/':

if (num2 != 0)

result = num1 / num2;

else {

cout << "错误：除数不能为0！" << endl;

return 0;

}

break;

case '%':

if (num2 != 0)

result = (int)num1 % (int)num2;

else {

cout << "错误：除数不能为0！" << endl;

return 0;

}

break;

default:

cout << "错误：无效的运算符！" << endl;

return 0;

}

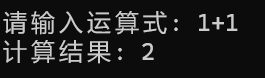
cout << "计算结果: " << result << endl;

return 0;

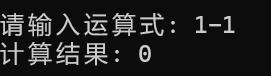
}

运行结果：

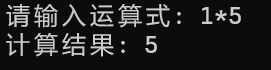
加法：



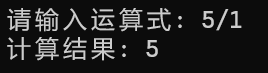
减法：



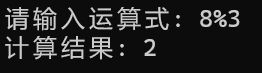
乘法：



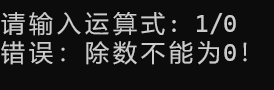
除法：



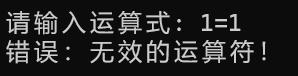
取余：

、

除数为0：



运算符非法：



基于堆栈的计算器代码：

#include <iostream>

#include <stdlib.h>

#include <math.h>

#include <string>

using namespace std;

# define numMaxSize 15

// 定义结构体的模板

template<typename T>

struct Stack

{

// 定义堆栈结构

T\* top;// 指向栈顶

T\* base;// 指向栈底

int stackSize;// 栈的大小

};

// 定义函数的模板

template<typename T>

void initStack(Stack<T>\* stack)// 栈的初始化

{

stack->base = stack->top = new T[1024];// 默认定义栈的大小为1024

stack->stackSize = 1024;

}

template<typename T>

void push(Stack<T>\* stack, T e) //将元素e压入栈顶

{

\*stack->top = e;

stack->top++;// 栈顶指向下一元素位置

}

template <typename T>

int pop(Stack<T>\* stack, T\* e)// 将元素e弹出栈

{

// 若栈顶和栈底位置相同则说明栈空

if (stack->base == stack->top)

{

return 0;

}

// 栈非空，则栈顶后移一位，并返回弹出元素

\*(stack->top--);

\*e = \*stack->top;

return 1;

}

template <typename T>

int isEmpty(Stack<T>\* stack)// 判断栈是否为空

{

if (stack->top == stack->base)

{

return true;

}

else

return false;

}

template <typename T>

int getTop(Stack<T>\* stack, T\* e)//将栈顶元素赋给e

{

if (isEmpty(stack))

{

return false;

}

\*e = \*(stack->top - 1);

}

// 中缀式转后缀式

char\* MidToAfter(const char\* mid)

{

char\* after = new char[1024]; // 存储字符的后缀式

memset(after, 0, 1024);// 将后缀式元素设为0，便于后续识别

int afterIndex = 0;//后缀式的下标

int midIndex = 0;// 中缀式的下标

char midChar; // 中缀式的字符

Stack<char> stack;// 存储中缀式的符号的栈

initStack(&stack);

int numIndex = 0;// 存储数字的下标

bool flag = true;// 判断小数点个数，若超过一个则报错

int numPoint = 0;// 记录小数点个数

// 开始转换

while (mid[midIndex])

{

midChar = mid[midIndex++];

if (midChar == '\n')break;// 若为回车键则直接结束循环

else if (midChar == ' ')continue;// 若为空格键则继续读入下一个字符串

else if (midChar <= '9' && midChar >= '0')// 若为数字则将其存入num和after

{

flag = true;

numPoint = 0;

while (((midChar <= '9' && midChar >= '0') || midChar == '.') && numIndex < numMaxSize)

{

if (midChar == '.') numPoint++;

if (numPoint > 1)

{

cout << "您的输入有误，错误为小数点个数大于1" << endl;

return NULL;

}

numIndex++;

after[afterIndex++] = midChar;// 将数字入后缀式

midChar = mid[midIndex++];// 读入后续中缀式数字位

}

after[afterIndex++] = ' ';// 读完一个数后，后加上空格

}

if (midChar == '\n')break;// 在读入数字后，再读入字符，若为回车键则直接结束循环

else if (midChar == ' ')continue;// 在读入数字后，再读入字符，若为空格键则继续读入下一个字符串

else if (midChar < '0' || midChar >'9')// 读入字符，若为运算符则将运算符入stack栈

{

numIndex = 0;

// (和^，优先级最高，直接入栈

if (midChar == '(' || midChar == '（' || midChar == '^')push(&stack, midChar);

// 读入)，则将括号内所有符号弹出栈，并入后缀式

else if (midChar == ')' || midChar == '）')

{

bool flag1 = true;

while (!isEmpty(&stack))

{

pop(&stack, &midChar);

if (midChar == '(' || midChar == '（')

{

flag1 = false;

break;

}

after[afterIndex++] = midChar;

after[afterIndex++] = ' ';

}

if (flag1)

{

cout << "您的输入有误：缺少括号";

return NULL;

}

}

// 读入+和-

else if (midChar == '+' || midChar == '-')

{

char element;

getTop(&stack, &element);// 获取栈顶元素

// 若栈空或只有左括号，则将+或-入栈

if (isEmpty(&stack) || element == '(' || element == '（') push(&stack, midChar);

// 否则将栈内除左括号以外的符号弹出栈，入后缀式中

else

{

while (!isEmpty(&stack))

{

pop(&stack, &element);

// 若弹出元素为左括号，则放回栈中，并结束循环

if (element == '(' || element == '（')

{

push(&stack, element);

break;

}

after[afterIndex++] = element;// 将弹出元素入后缀式

after[afterIndex++] = ' ';

}

push(&stack, midChar);//最后将+或-入栈

}

}

// 读入\*和/和%

else if (midChar == '\*' || midChar == '/' || midChar == '%')

{

char element;

getTop(&stack, &element);

// 若栈空或只有左括号或+和-，则将\*或/或%入栈

if (isEmpty(&stack) || element == '(' || element == '（' || element == '+' || element == '-') push(&stack, midChar);

// 否则将栈内除左括号以外的符号弹出栈，入后缀式中

else

{

while (!isEmpty(&stack))

{

pop(&stack, &element);

if (element == '(' || element == '（' || element == '+' || element == '-')

{

push(&stack, element);

break;

}

after[afterIndex++] = element;

after[afterIndex++] = ' ';

}

push(&stack, midChar);

}

}

// 否则输入错误

else

{

cout << "您的输入有误！" << endl;

return NULL;

}

}

}

// 转换结束，将符号栈中符号弹出，入后缀式

char element;

while (!isEmpty(&stack))

{

pop(&stack, &midChar);

after[afterIndex++] = midChar;

after[afterIndex++] = ' ';

}

return after;

}

// 利用后缀式进行计算

double calculate(const char\* after)

{

if (after == NULL) return NULL;

int afterIndex = 0;// 后缀式下标

double number\_1 = 0, number\_2 = 0;// 计算两个数

char afterChar;// 存储后缀式中的字符

Stack<double> jisuan;// 用于计算的栈

initStack(&jisuan);

char num[numMaxSize];// 用于读取一个数

int numIndex = 0;// num的下标

// 判断中缀式是否正确

for (int i = 0; after[i] != 0; i++)

{

if (after[i] == '(' || after[i] == '（')

{

cout << "请输入右括号" << endl;

return NULL;

}

}

while (after[afterIndex])

{

afterChar = after[afterIndex++];// 读入后缀式的字符串

// 读入为数字

if (afterChar >= '0' && afterChar <= '9')

{

numIndex = 0;

while (((afterChar >= '0' && afterChar <= '9') || afterChar == '.') && numIndex < numMaxSize)

{

num[numIndex++] = afterChar;

afterChar = after[afterIndex++];

}

num[numIndex] = 0;//表示读入了整个数字

double number = atof(num);// 将字符串转为数字

push(&jisuan, number);

}

// 读入为符号

if (afterChar < '0' || afterChar > '9')

{

numIndex = 0;

if (afterChar == ' ') continue;// 空格则读入下一个字符

else

{

double result = 0;//记录运算结果

switch (afterChar)

{

case'+':

pop(&jisuan, &number\_1);

pop(&jisuan, &number\_2);

result = number\_1 + number\_2;

push(&jisuan, result);

break;

case'-':

pop(&jisuan, &number\_1);

pop(&jisuan, &number\_2);

result = number\_2 - number\_1;

push(&jisuan, result);

break;

case'\*':

pop(&jisuan, &number\_1);

pop(&jisuan, &number\_2);

result = number\_1 \* number\_2;

push(&jisuan, result);

break;

case'/':

pop(&jisuan, &number\_1);

pop(&jisuan, &number\_2);

if (number\_1 == 0)

{

cout << "错误：除数为0" << endl;

break;

}

result = number\_2 / number\_1;

push(&jisuan, result);

break;

case'%':

pop(&jisuan, &number\_1);

pop(&jisuan, &number\_2);

result = int(number\_2) % int(number\_1);

push(&jisuan, result);

break;

case'^':

pop(&jisuan, &number\_1);

pop(&jisuan, &number\_2);

result = pow(number\_2, number\_1);

push(&jisuan, result);

break;

default:

return NULL;

break;

}

}

}

}

if (isEmpty(&jisuan))

{

cout << "您的表达式输入错误" << endl;

return NULL;

}

pop(&jisuan, &number\_1);

return number\_1;

}

void main()

{

cout << "请输入计算式：";

char\* mid = new char[1024];

fgets(mid, 1024, stdin);//可读入\n,用以标记字符串输入结束

char\* after = MidToAfter(mid);

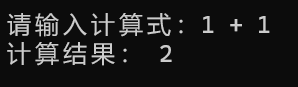
double result = calculate(after);

cout << "计算结果： " << result << endl;

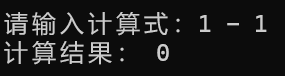
}

运行结果

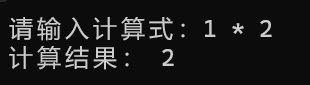
加操作：



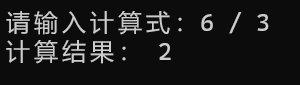
减操作：



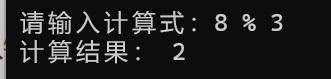
乘操作：



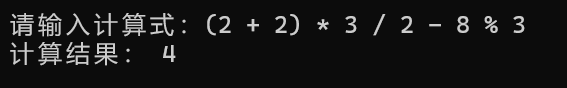
除操作：



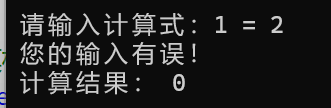
取余：

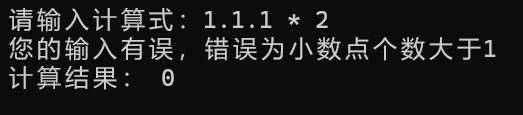


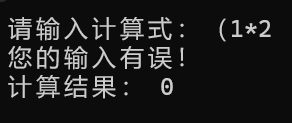
复合运算：



运算符非法：







#### 5．

代码：

#include <iostream>

using namespace std;

int main\_5() {

char c;

int letter\_count = 0;

int space\_count = 0;

int digit\_count = 0;

int other\_count = 0;

cout << "请输入一行字符：\n";

while ((c = getchar()) != '\n') {

if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')) {

letter\_count++;

}

else if (c == ' ') {

space\_count++;

}

else if (c >= '0' && c <= '9') {

digit\_count++;

}

else {

other\_count++;

}

}

cout << "英文字母个数：" << letter\_count << endl;

cout << "空格个数：" << space\_count << endl;

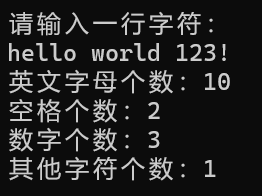
cout << "数字个数：" << digit\_count << endl;

cout << "其他字符个数：" << other\_count << endl;

return 0;

}

运行结果：



#### 6.

代码：

#include <iostream>

using namespace std;

// 计算最大公约数

int gcd(int a, int b) {

if (b == 0) {

return a;

}

return gcd(b, a % b);

}

// 计算最小公倍数

int lcm(int a, int b) {

return (a \* b) / gcd(a, b);

}int main() {

int a, b;

cout << "请输入两个正整数：";

cin >> a >> b;

// 调用函数计算最大公约数和最小公倍数

int greatestCommonDivisor = gcd(a, b);

int leastCommonMultiple = lcm(a, b);

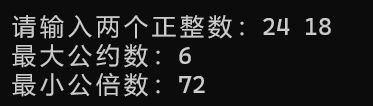
cout << "最大公约数：" << greatestCommonDivisor << endl;

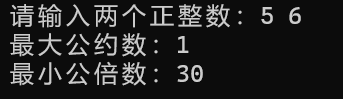
cout << "最小公倍数：" << leastCommonMultiple << endl;

return 0;

}

运行结果：





#### 7.

代码：

#include <iostream>

using namespace std;

int main() {

for (int i = 1; i < 6; i++) {

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

cout << '\*';

}

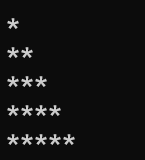
cout << endl;

}

return 0;

}

运行结果：



#### 8.

代码：

#include <iostream>

#include <cmath>

using namespace std;

double sqrtByIteration(double a) {

double x = a; // 初始猜测值

double epsilon = 1e-5; // 精度控制，小于10^-5

double diff = epsilon + 1; // 初始差异

while (diff > epsilon) {

double nextX = 0.5 \* (x + a / x); // 使用牛顿迭代法求下一个估计值

diff = abs(nextX - x); // 计算差异

x = nextX; // 更新估计值

}

return x;

}

int main() {

double a;

cout << "请输入一个大于0的数：";

cin >> a;

while (a <= 0) {

cout << "您的输入有误，请重新输入一个大于0的数：";

cin >> a;

}

double sqrtValue = sqrtByIteration(a);

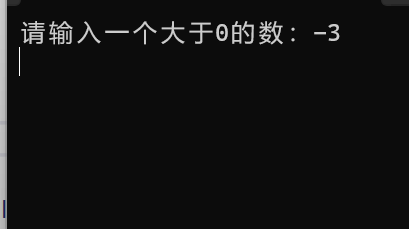
cout << a << " 的平方根为: " << sqrtValue << endl;

return 0;

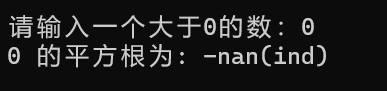
}

运行结果：

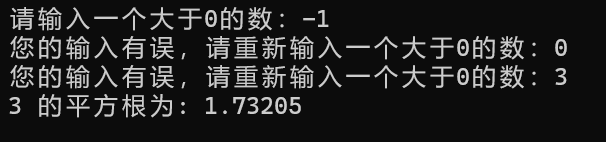
输入负值，程序进入死循环



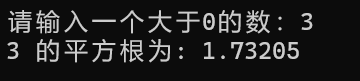
输入0



修改为无法输入非正值



修改精度为10-10



#### 9.

9.1 某一天购买的苹果数量不超过100的最大数

代码：

#include <iostream>

#include <math.h>

using namespace std;

int main() {

int count = 0;

int total = 0;

int i = 1;

double avg = 0;

while (true) {

// 今天买的个数

count = pow(2, i);

if (count >= 100) {

i--;

avg = total \* 0.8 / i;

break;

}

// 总共买的个数

total += count;

cout << "第" << i << "天买了" << count << "个苹果" << endl;

i++;

}

cout << "平均花销：" << avg << endl;

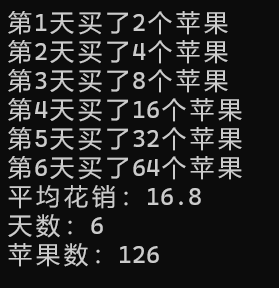
cout << "天数：" << i << endl;

cout << "苹果数：" << total << endl;

return 0;

}

运行结果：



9.2 总共购买的苹果数量不超过100的最大数

代码：

#include <iostream>

#include <math.h>

using namespace std;

int main() {

int count = 0;

int total = 0;

int i = 1;

double avg = 0;

while (true) {

// 今天买的个数

count = pow(2, i);

// 总共买的个数

total += count;

if (total >= 100) {

i--;

total -= count;

avg = total \* 0.8 / i;

break;

}

cout << "第" << i << "天买了" << count << "个苹果" << endl;

i++;

}

cout << "平均花销：" << avg << endl;

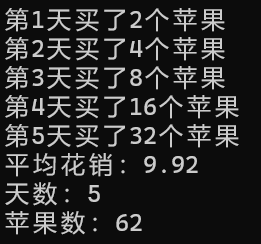
cout << "天数：" << i << endl;

cout << "苹果数：" << total << endl;

return 0;

}

运行结果：



**四、遇到的问题与解决方法**

对于第二题，考虑到x的取值范围为(0, 10)，于是我在编写代码时，即判断输入是否在0~10，若不在则请用户重新输入。

对于第四题，考虑到了两层实现：首先是直接读取用户输入的数字与运算符，并利用switch判定运算符是否非法，但因为灵活性较低，仅能进行一次简单的运行，考虑到第二层实现。

因为考虑到计算器要求可用括号等运算符来表示先后顺序，所以不能直接用fgets方法读取数据后逐字符分析。我选择使用栈来存储用户输入的运算符，并将中缀式转化为后缀式进行输出。

首先通过设计堆栈，设置栈顶、栈底和栈的大小，再完成栈的各个操作，如元素入栈、弹出栈，栈的初始化，判断栈是否为空和获得栈顶元素。

再将用户输入的中缀式以字符的形式读入：①若读入为数字，则将数字以字符形式放入后缀式的字符数组中，直到读入不为数字或小数点；②若读入为符号，则先将符号入栈，根据后续读入符号的优先级，选择是否将符号从栈弹出，入后缀式。如1\*2+3，首先将“1”入后缀式，“\*”入栈，“2”入后缀式，此时“+”的优先级小于栈中的“\*”，故将“\*”弹出，入后缀式，再将“+”入栈，3入后缀式，最后将栈中元素弹出，入后缀式；③若读入为空格，则再读入下一个字符；④若读入为回车”\n”，则表示转换结束。（每个后缀式中元素会以空格隔开，以便后续计算）。

对于后缀式的计算比较简单，首先将后缀式中两个数字（此时为字符串），转为浮点型并入栈，再继续读入后缀式中元素，若为数字则继续入栈，若为符号则在占中弹出两个数字并计算，将计算结果入栈。

对于第五题，主要是对于读取空格与’\n’，我使用了getchar()方法，当读取到’\n’时，停止读取，若为空格，则继续读取到下一位。

回答题目8的问题2：

当修改精度为10-10时，程序并不会报错，但输出的结果并没有显著变化，我认为由于在计算机表示浮点数时，存在着舍入误差和数值精度限制，从而导致无法达到完全精确的计算。

而且，浮点数的表示范围也是有限的。当计算结果非常接近0时，精度要求越高，可能会出现下溢错误。这意味着计算结果变得非常接近0，无法再继续更新，从而无法满足更严格的精度要求。

对于第九题，考虑到问题可以理解为总数不超过100的最大整数，或是最后一天购买的数量不超过100的最大整数，我分别给出了代码和答案。

**五、体会**

这次的实验内容相对较多，但每一题都让我收获颇多。第一题代码量虽较小，但既涉及到了大小写字母ascii码值相差32的概念，又有将字符直接转化为ascii编码的部分。第四题，我又利用起来堆栈的知识做出了更好的回答，第五题又考虑到了读取空格与字符串的各个方法，第六题又回忆起了最大公约数与最小公倍数的计算公式，第七题我本想使用cout<<’\*’\*num<<endl;的方法进行输出，后来才发现c++并不支持这样的方法，于是我又换了使用两层嵌套，第八题又涉及到了迭代的概念。通过这次实验，我相信我的C++编码能力又能得到进一步提升！