程序设计基础第一次上机实验报告

作者：吴靖宇 日期：2021-10-13

1.实验目的与要求

（1） 使用C++语言编写简单的计算机程序，掌握C++程序从编辑、编译到运行的全过程，初步了解C++程序的特点。

（2） 掌握C++语言的数据类型（包括对各种数据类型变量的定义、初始化、赋值等）、运算符和表达式的使用。

（3） 掌握数据的输入输出方法。

（4） 理解基本的解题技巧，掌握用自顶向下、逐步求精的过程设计算法。

（5） 熟练掌握if/else语句和switch语句的使用。

（6） 熟练掌握while语句、do/while语句和for语句的使用以及区别。

（7） 掌握break和continue程序控制语句的使用。

（8） 理解并掌握如何模块化的用函数来构建程序。

（9） 掌握定义函数的方法，了解函数的重载方法，了解内联函数的机制。

（10） 掌握函数间传递信息的机制。

（11） 掌握函数的嵌套调用（和递归调用）的方法。

2.实验环境

OS：Win10 家庭版

IDE：Visual Studio 2019

3.实验内容

1.代码测试

1.

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

int main()
{
    char ch;
    cin >> ch ;
    ch = ( ch >= 'A' && ch <= 'Z' ) ? ( ch + 32 ) : ch ;
    ch = ( ch >= 'a' && ch <= 'z' ) ? ( ch - 32 ) : ch ;
    cout << ch << endl;

    return 0;

}
```

输入 1：

```
A
```

输出 1：

```
A
```

输入 2：

```
q
```

输出 2：

```
Q
```

输入 3：

```
4
```

输出 3：

```
4
```

结果分析：

本程序用了两个条件运算符来决定输出结果：

```
ch = ( ch >= 'A' && ch <= 'Z' ) ? ( ch + 32 ) : ch ;

ch = ( ch >= 'a' && ch <= 'z' ) ? ( ch - 32 ) : ch ;
```

目的是：输入大写字母，返回小写字母给原来的的变量

但是根据输入1，我得到了意想不到的的结果，发现删除第二行代码后，输出又发生变化

输入 4：

```
A
```

输出 4：

```
a
```

原因分析：第二行代码把第一行转成小写字母的ch，又变成大写字母了

解决方案：可以使用下面的代码

```
ch = ( ch >= 'A' && ch <= 'Z' ) ? ( ch + 32 ) :

 ( ch >= 'a' && ch <= 'z' ) ? ( ch - 32 ) : ch ;
```

执行完第一次判断后，继续第二次判断，复现输入1,4，结果均为小写。

反思：在遇到较为复杂的判断条件时，建议避免使用条件运算符

2.

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

int main()
{
    int m;
    float x;
    bool bi, br;

    cout << "\n int m=";
    cin >> m;
    bi = m > 1000;
    cout << "\n float x=";
    cin >> x;
    br = x <= 1e3;
    cout << bi << ',' << br << endl;//判断输入的整数和浮点数是否大于一千或小于1e3，并返回判断结果的布尔值

    return 0;
}
```

输入1：

`100 40`

输出1：

`0,1`

输入2：

`2000  3000`

输出2：

`1, 0`

输入3：

`1000  1000`

输出3：

`0,1`

输入4：

`2000  300`

输出4：

`1,1`

输入5：

`100 4000`

输出5：

`0,0`

功能分析：判断输入的整数和浮点数是否大于一千或小于1e3，并返回判断结果的布尔值

知识点：在c++中科学计数法的表示方法“A e B”相当于：

$$A * 10^B$$

其中A和B均为有符号整型。

3.

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

int main()
{
    int n;
    cin >> n;
    if (n++ < 10)
        cout << n << endl;
    else
        cout << n-- << endl;
        //cout<<n;
    return 0;
}
```

输入1：`8`

输出1：`9`

n=9

输入2：`9`

输出2：`10`

n=10

输入3：`10`

输出3：`11`

n=10

输入4：`11`

输出4：`12`

n=11

功能分析：连加（减）符号的前置后置，对变量的影响：

条件判断句中的n++实质上是：

`n<10;n=n+1;`

else代码作用域中n--实质上是：

`cout<<n;n=n-1;`

如果运算符前置，则会先执行递增（减）操作，然后再执行变量的其他操作。

4.

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

int  main()
{
    int m, n;
    m = 1000;
    n = 850;
    cout << "\n(" << m << ',' << n << ')';
    while (m != n) {
        while (m > n) {
            m = m - n;
            cout << '(' << m << ',' << n << ')';
        }
        while (m < n) {
            n = n - m;
            cout << '(' << m << ',' << n << ')';
        }
    }
    cout << "\n" << m;
    return 0;
}
```

功能分析：用广义欧几里得除法（辗转相除法）求1000和850的最大公因数，并返回计算过程

输出

```
(1000, 850)(150, 850)(150, 700)(150, 550)(150, 400)(150, 250)(150, 100)(50, 100)
(50, 50)
50
```

此外求最大公因数的方法还有穷举法、更相减损法、Stein算法

穷举法：

```
int divisor1(int a, int b) {
    int temp;
    temp = (a > b) ? b : a;   //求较小值
    while (temp > 0) {
        if (a % temp == 0 && b % temp == 0) {
            break;
        }
        else {
            temp--;
        }
    }
    return (temp);
```

```
//更相减损法求最大公约数
int gcd1(int a, int b) {
    int i = 0, temp, x = 0;
    while (a % 2 == 0 && b % 2 == 0) {    //m,n有公约数2时
        a /= 2;
        b /= 2;
        i += 1;
    }
    //a,b的值互换
    if (a < b) {
        temp = a;
        a = b;
        b = temp;
    }
    while (x) {
        x = a - b;
        a = (b > x) ? b : x;
        b = (b < x) ? b : x;
        if (b == (a - b)) {    //差和减数相等
            break;
        }
    }
    if (i == 0) {
        return b;
    }
    else {
        return (int)pow(2, i) * b;
    }
}
```

Stein算法：

```
//Stein算法函数非递归调用求最大公约数
int Stein(unsigned int x, unsigned int y) {
    int factor = 0;   //计数器
    int temp;

    //大数赋给x，小数赋给y
    if (x < y) {
        temp = x;
        x = y;
        y = temp;
    }
    if (0 == y) {
        return 0;
    }
    while (x != y) {
        if (x & 0x1) {
            if (y & 0x1) {   //x,y都为奇数
                y = (x - y) >> 1;
                x -= y;
            }
            else {    // x为奇数，y为偶数
                y >>= 1;
            }
        }
```

```
        else {
            if (y & 0x1) {   // x为偶数，y为奇数
                x >>= 1;
                if (x < y) {
                    temp = x;
                    x = y;
                    y = temp;
                }
            }
            else {   //x,y均为偶数
                x >>= 1;
                y >>= 1;
                ++factor;
            }
        }
    }
    return (x << factor);
}
```


5.

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

int  main()
{
    int m, n, k;

    m = 1000;
    n = 45;
    cout << "\n(" << m << ':' << n << ')';
    k = 0;
    while (m >= n) {
        m = m - n;
        k = k + 1;
    }
    cout << k << "---" << m << endl;

    return 0;
}//整除并求余数
```

功能分析：将两数相除，并返回商和余数

输出：

```
(1000:45)22---10
```

6.

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

int  main()
{
    int i;
    for (i = 1; i <= 5; i++) {
        if (i % 2)//i=1,3,5

            cout << '*';
        else
            continue;
        cout << '#';
    }
    cout << "$\n";

    return 0;
}
```

输出

`*#*#*#$`

i=1，3，5（奇数）输出*#

continue语句重开循环，不会输出#

7.

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

int  main()
{
    int a = 1, b = 10;

    do {
        b -= a;
        a++;
    } while (b-- <= 0);
    cout << "a=" << a << ",  b=" << b << endl;

    return 0;
}//a=2,b=8
```

输出：

`a=2,b=8`

do-while：会强制执行一次语法快中的内容，然后判断是否退出

8.

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

int hcf(int u, int v)
{
    int a, b, t, r;
    if (u > v) {
        t = u;
        u = v;
        v = t;
    }
    a = u;
    b = v;
    while ((r = b % a) != 0) {
        b = a;
        a = r;
    }
    return a;
}//最大公因数

int lcd(int u, int v, int h)
{
    return (u * v / h);
}//最小公倍数

int  main()
{
    int u, v, h, l;

    cin >> u >> v;
    h = hcf(u, v);
    cout << "hcf=" << h << endl;
    l = lcd(u, v, h);
    cout << "lcd=" << l << endl;
    return 0;
}
```

输入:

```
120 450
```

输出:

```
hcf = 30
lcd = 1800
```

公式:

$$gcd(a,b) = (a*b)/[a,b]$$

求最大公因数用的是广义欧几里得除法

9.

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

long fib(int g)
{
    switch (g) {
    case 0: return 0;
    case 1:
    case 2: return 1;
    }
    return (fib(g - 1) + fib(g - 2));
}

int  main()
{
    for (int i = 1; i <= 20; i++) {
        cout << fib(i) << " ";
    }
    return 0;
}
```

输出：

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
```

功能分析：用函数迭代法求斐波那契数列前n项

用这种方法可以求绝大多数可以写出迭代式和通项公式的求和或求积

10.

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

int k = 1;

void fun(int m)
{
    m += k;
    k += m;
    {
        char k = 'B';
        cout << "(2)" << char(k - 1) << endl;
    }
    cout << "(3)" << m << ',' << k << endl;
}

int  main()
{
    int i = 4;
  //int fun (int);
    fun(i);
    cout << "(1)" << i << ',' << k << endl;
    return 0;
}
```

输出：

```
(2)A
(3)5,6
(1)4,6
```

知识点：变量的作用域

在全局域中，k为int型，值为1，在fun之中，代码块之外，k为int型，代码块中，k被重定义为char型，在main函数中，k为int型

11.

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

void subp()
{
    static int x = 0, y = 0; //问题（1），静态变量内存地址不变，其值不会随着函数调用改变
    int a = 1, b = 1;
    a = a + x;
    b = b + y;
    cout << "subp函数输出：\n";
    cout << setw(5) << a << setw(5) << b << '\n';
    cout << setw(5) << x << setw(5) << y << '\n';//setw控制输入的间隔
    x++;
    y++;
}

int x, y;

int  main()
{
    int a = 9, b = 3;
    x = a - b;
    y = a + b;
    subp();
    cout << "main函数输出：\n";
    cout << setw(5) << a << setw(5) << b << '\n';
    cout << setw(5) << x << setw(5) << y << '\n';
    subp();
    return 0;
}
```

输出：

```
subp函数输出：
    1    1
    0    0
main函数输出：
    9    3
    6   12
subp函数输出：
    2    2
    1    1
```

代码分析：

1.setw()

是iomanip库中的函数，主要是为了设定空格

2.静态变量

静态变量内存地址不变，其值不会随着函数调用改变，不会因为重定义，而导致x，y值发生改变。

2.思考题

1.用cmath实现相应函数

1.3：实现

$$sinX + cosX + arctanX$$

```cpp
#include <iostream>
#include <cmath>
using namespace std;
const double PI = acos(-1);
int main()
{

    double x;
    cin >> x;
    cout << sin(x) + cos(x) + atan(x);
    return 0;
}
```

1.4：实现

$$e^{x+y} + e^{x-y}$$

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double x, y;
    cin >> x >> y;
    cout << exp(x + y) + exp(x - y);
    return 0;
}
```

2.对任意输入的四位整数，分别求出其各位数字，并按从后到前的顺序依次输出。例如，输入为1234时，输出结果为4，3，2，1。

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

int main()
{
    unsigned int n, N;
    int a[20];
    cin >> n;
    int i = 0;
    while (n > 0)
    {
        a[i] = n % 10;
        i++;
        n /= 10;
    }
    N = 0;
    i = i - 1;
    int k = 1;
    while (i >= 0)
    {

        N += (a[i] * k);
        k *= 10;
        i--;
    }
    cout << N <<" ";

    return 0;

}
```

将每一位数通过不断余10，将每一位数存入数组，用计数器计算有多少位，并从数组反向输出，i递减到0时，停止输出。

3.求解下面函数的值。*<3><4>选做一题*

$$e^{x+y}(x<0,y<0)$$

$$z=ln(x+y)(1\le x+y<10)$$

$$log_{10}|x+y|+1,else$$

```
#include<iostream>
#include <cmath>
using namespace std;
int main()
{
    double x, y;
    cin >> x >> y;
    if(x<0&&y<0)
    {
        cout << exp(x + y);
    }
    else if (x+y>=1&&x+y<10)
    {
        cout<<log(x + y);
    }
    else
    {
        cout<<log10(fabs(x + y) + 1);
    }
    return 0;
}
```

fab取绝对值

5.

2.

```
#include <iostream>
using namespace std;
long fact(int a)
{
    long t=1;
    long s = 0;
    for (int i=1;i<=a;i++)
    {
        t = t * i;
        s += t;
    }
    return s;
}
int main()
{
    int a=7;
    cout<<fact(a);
```

```
}
```

迭代公式：

$$a_n = a_{n-1} + (n-1)! * n$$

3.

```
#include <iostream>
using namespace std;
int main()
{
    int sum = 0;
    for (int i = 1; i < 40;i+=2)
    {
        sum += i * (i + 1);
    }
    cout << sum;
    return 0;
}
```

迭代公式：

$$s_n = s_{n-1} + n * (n-1)$$

6.生成数字金字塔

```
#include <iostream>
using namespace std;
int main()
{
    for(int i=0;i<=10;i++)
    {

        for(int a=0;a<10-i;a++)
        {
            cout << "  ";
        }
        for(int n=1;n<=2*i+1;n+=2)
        {
            cout << n << ' ';
        }
        int n = 2 * i + 1;
        while(n>1)
        {
            n -= 2;
            cout << n<<' ';
        }
        cout << endl;
    }
    return 0;
}
```

输出：

生成算法设计：

1.生成左侧树，设定每行的输出上限：2n+1

2.生成右侧树：用2n+1递减，生成右侧树

3.生成空格：生成这一行的空格缩进，将树向右推

4.循环迭代至中央数为21

7.编程产生出1到10以内的所有数对<i,j>并输出,其中i>j

```
#include <iostream>
using namespace std;
int main()
{
    for(int i=1;i<=10;i++)
    {
        for (int j =1;j<=10;j++)
        {
            if (i>j)
            {
                cout << '<' << i << ',' << j << '>' << ' ';
            }

        }
    }
    return 0;
}
```

用i,j双重遍历,当i>j输出相关数对:

```
<2,1> <3,1> <3,2> <4,1> <4,2> <4,3> <5,1> <5,2> <5,3> <5,4> <6,1> <6,2> <6,3>
<6,4> <6,5> <7,1> <7,2> <7,3> <7,4> <7,5> <7,6> <8,1> <8,2> <8,3> <8,4> <8,5>
<8,6> <8,7> <9,1> <9,2> <9,3> <9,4> <9,5> <9,6> <9,7> <9,8> <10,1> <10,2> <10,3>
<10,4> <10,5> <10,6> <10,7> <10,8> <10,9>
```

算法改进:

```
#include <iostream>
using namespace std;
int main()
{
    for(int i=1;i<=10;i++)
    {
        for (int j =1;j<i;j++)
        {
                cout << '<' << i << ',' << j << '>' << ' ';
        }
    }
    return 0;
}
```

8.编程求出10000以内的所有符合如下条件的数：其高位数字小于低位数字。

如12，238，3578等。但21，548不符合条件。

```
nclude <iostream>
using namespace std;
int main()
{
    int a[5];
    int temp;
    for(int n =10;n<=10000;n++)
    {
        temp = n; int i = 0;
        while(temp!=0)
        {

            a[i] = temp % 10;
            i++;
            temp /= 10;

        }
        for(int j=0;j<i;j++)
        {
            if (a[j] <= a[j + 1])
            {
                break;
            }
            if(j+1==i)//判断结束，且符合条件时，输出该数
            {
                cout << n << ' ';
            }

        }

    }
}
```

```
12 13 14 15 16 17 18 19 23 24 25 26 27 28 29 34 35 36 37 38 39 45 46 47 48 49 56
57 58 59 67 68 69 78 79 89 123 124 125 126 127 128 129 134 135 136 137 138 139
145 146 147 148 149 156 157 158 159 167 168 169 178 179 189 234 235 236 237 238
239 245 246 247 248 249 256 257 258 259 267 268 269 278 279 289 345 346 347 348
349 356 357 358 359 367 368 369 378 379 389 456 457 458 459 467 468 469 478 479
489 567 568 569 578 579 589 678 679 689 789 1234 1235 1236 1237 1238 1239 1245
1246 1247 1248 1249 1256 1257 1258 1259 1267 1268 1269 1278 1279 1289 1345 1346
1347 1348 1349 1356 1357 1358 1359 1367 1368 1369 1378 1379 1389 1456 1457 1458
1459 1467 1468 1469 1478 1479 1489 1567 1568 1569 1578 1579 1589 1678 1679 1689
1789 2345 2346 2347 2348 2349 2356 2357 2358 2359 2367 2368 2369 2378 2379 2389
2456 2457 2458 2459 2467 2468 2469 2478 2479 2489 2567 2568 2569 2578 2579 2589
2678 2679 2689 2789 3456 3457 3458 3459 3467 3468 3469 3478 3479 3489 3567 3568
3569 3578 3579 3589 3678 3679 3689 3789 4567 4568 4569 4578 4579 4589 4678 4679
4689 4789 5678 5679 5689 5789 6789
```

9.求任一整数N的标准分解式，即素数因子之积。

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int N;
    cin >> N;
    int a[20];
    int i = 2;
    while (N/i!=0)
    {
        if(N%i==0)
        {

            N /= i;
            cout << i << ' ';
            i = 2;//重置因数
        }
        else
        {
            i++;
        }

    }


}
```

输入：

```
20
```

输出：

```
2 2 5
```

4.实验总结：

1.减少使用条件运算符

2.有规律的数字，图形输出，可以借用循环的嵌套

3.时刻检查计数器在循环体中的位置

4.cmath中有许多的有用数学函数

5.参考资料：

(57条消息) LaTeX 中插入数学公式*nowting*的博客-*CSDN*博客latex数学公式

(57条消息) cmath(常用函数)*wangtao*的博客-*CSDN*博客cmath

(57条消息) 四种求最大公约数算法*hi*你好-CSDN博客_求最大公约数的方法有哪些