C/C++ 程序设计综合训练设计说明书

（项目二）

目 录

一、项目总体设计方案

1. 系统功能需求分析

复数计算器采用命令行交互方式运行，围绕两个当前操作数 A 与 B 进行计算。程序启动后，先给出输入规则提示，随后依次读取 A、B，并进入循环菜单。菜单界面需要在每轮操作前显示当前 A 与 B 的值，便于用户在多次运算过程中确认对象与结果来源。

输入方面，支持常见复数表示的录入，包括 $a+b$i、$a-b$i、i、$-$i、纯实数（如 5）以及只含虚部的形式（如 3i）。输入过程具备基本的格式约束：除数字、小数点、符号 `+/-` 与字母 `i` 外，其它字符应判为非法；若出现 `i`，则要求它只允许出现在末尾，避免诸如 `2i3` 之类的歧义表达。对于不符合格式的输入，系统应给出提示并要求重新输入，直到获得有效复数为止。

运算方面，提供针对 A 与 B 的四则运算：加、减、乘、除。每次运算需要以“表达式 + 结果”的形式输出。除法运算需要考虑被除数为零复数的情况：当 B 的模长为零时，除法应被禁止并输出明确提示，以避免产生无意义结果或运行异常。

交互与健壮性方面，系统必须具有循环菜单，用户可重复选择功能而无需重启程序；同时需要处理菜单输入的异常情况（例如误输入字符导致无法读取整型选项），此时应清理输入流状态并返回菜单重新选择。退出操作应提供明确的结束提示，并在需要的环境下避免窗口瞬间关闭，保证用户能够看到最终输出。

2. 系统整体结构设计

如图 1，系统核心数据由 `Complex` 类承载，交互与流程由主控循环驱动。整体数据流为：输入得到 $A, B \rightarrow$ 在内存中保持为对象 $\rightarrow$ 根据菜单选择触发运算 $\rightarrow$ 输出结果并返回菜单。

数据对象层由 `Complex` 类构成，内部包含两个双精度数据成员表示实部与虚部，并提供带默认参数的构造函数，用于完成对象初始化。类还提供模长计算接口，为除法合法性判断提供基础。

运算能力层主要通过运算符重载实现：四则运算符 `+/-/*//` 负责完成复数之间的代数运算；流运算符 `<<` 将复数以规范形式输出到终端，处理 0、纯实数、纯虚数以及虚部为 ± 1 等边界显示；流运算符 `>>` 负责从输入流读取一整行并完成字符串解析，将输入形式映射为实部与虚部的数值。为了支撑输入解析，系统额外设置了数值字符串合法性检查逻辑，用于区分可被解释为数字的片段与非法片段。

交互控制层由若干独立函数完成分工：输入函数负责“提示 + 读取 + 失败重试”的逻辑；菜单显示函数负责“状态展示 + 选项输出”；每个运算功能对应一个封装函数，用于调用运算符重载并按统一格式输出结果；主函数承担初始化、两次复数输入、循环菜单、分支调度与退出控制。

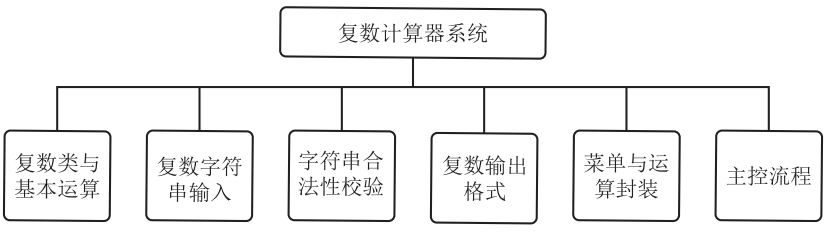

图 1 复数计算器系统功能模块图

二、项目详细设计方案

1. 复数类 Complex 与基本运算设计

类 Complex 用两个 double 型成员 real 和 imag 表示复数的实部与虚部，对应数学形式 $z = \text{real} + \text{imag}\,\text{i}$。构造函数 Complex(double r = 0.0, double i = 0.0) 通过默认参数支持直接生成 $0 + 0\text{i}$，同时也可以按给定实部、虚部初始化复数对象，后续所有运算都围绕这两个成员展开。

成员函数 getModulus() 返回复数的模长 $|z| = \sqrt{\text{real}^2 + \text{imag}^2}$，供除法运算前判断分母是否为零。

加法、减法、乘法和除法分别通过成员运算符重载实现：

- operator+ 实现

$$(a + b\text{i}) + (c + d\text{i}) = (a + c) + (b + d)\text{i},$$

逻辑上直接对 real 和 imag 做逐分量相加，生成局部对象并返回。

- operator- 实现

$$(a + b\text{i}) - (c + d\text{i}) = (a - c) + (b - d)\text{i},$$

对应实部和虚部分别做相减。

- operator* 实现

$$(a + b\text{i})(c + d\text{i}) = (ac - bd) + (ad + bc)\text{i},$$

先按复数乘法展开公式计算实部 $ac - bd$，再计算虚部 $ad + bc$，写入结果对象。

- operator/ 对应

$$\frac{a + b\text{i}}{c + d\text{i}} = \frac{(a + b\text{i})(c - d\text{i})}{c^2 + d^2} = \frac{ac + bd}{c^2 + d^2} + \frac{bc - ad}{c^2 + d^2}\text{i},$$

内部计算分母 temp = B.real*B.real + B.imag*B.imag,然后分别计算实部 (real*B.real + imag*B.imag)/temp 和虚部 (imag*B.real - real*B.imag)/temp。对分母为零的防护放在调用侧的 Div 函数，通过 getModulus() 判断，当前实现中不在运算符内部主动中止。

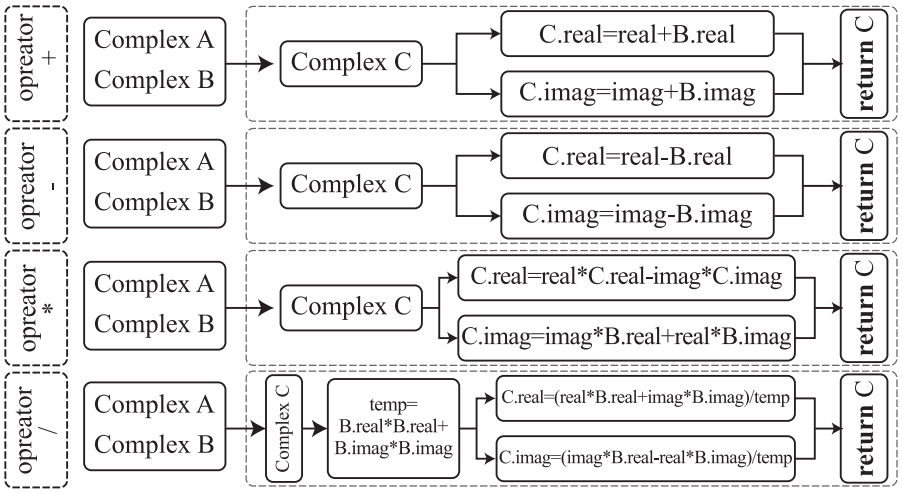

图 2 Complex 基本运算流程图

2. 字符串合法性校验——函数 IsValidNumber 的设计

IsValidNumber(string s) 用来判断一个字符串是否可以视为合法的浮点数形式，为复数解析函数提供基础检查。函数首先排除空串，其余情况遍历每个字符，只允许以下几类：

开头位置可以是 '+' 或 '-'，表示符号位；小数点 '.' 被允许出现，但累计出现次数超过 1 即视为非法；除上述三类字符外，必须是数字 '0'--'9'。一旦遇到其它字符，立即返回 false。

该函数不处理指数形式，只覆盖简单的带符号/小数点的十进制数，例如 "3"、"-2.5"、"+0.75"、".5" 等。复数解析时会多次调用这个函数，用于验证拆分后的实部和虚部系数是否合法。

3. 复数字符串输入解析——operator>> 的设计

输入运算符重载 istream& operator>>(istream& is, Complex& A) 的目标是支持常见复数写法，并在出现格式错误时通过 failbit 将错误状态传给上层交互逻辑。

读取阶段先通过 is.peek() 检查缓冲区首字符是否为换行，如果是则先忽略该换行，以消除上一次输入留下的影响。随后调用 getline 获得一整行字符串 strPtr 作为待解析对象。若整行为空，直接设置流状态为失败并返回，让外层逻辑决定是否重试。

格式检查和解析同时进行。函数遍历整行字符，记录第一个字符 'i' 出现的位置 i_pos，并在同一轮遍历中检查每个字符是否属于允许集合：数字、小数点、加号、减号或 'i'。一旦发现非法字符，立即设置失败状态并返回。遍历结束后，若出现了 'i' 但不在字符串末尾位置，则视为结构错误，同样标记失败。

根据是否含有 'i' 分两类处理。若不含 'i'，整体当作实数。此时调用 IsValidNumber 检查整行是否合法，合法则用 atof 转换为实部并将虚部设为 0，否则标记失败。

含有 'i' 时，会去掉结尾的 'i'，得到主体部分 temp。接下来从字符串尾部向前搜索分隔符号 '+' 或 '-'，只在索引大于 0 时认定为实部和虚部系数之间的界限，避免把开头的符号误当成分隔符。找到分隔符号时，将 temp 拆分为左侧的 s_real 和右侧的 s_imag；没有找到则认为输入是纯虚数形式，实部取 0，整个 temp 作为虚部系数字符串。

实部和虚部的赋值分别处理。实部部分中，空串或 "+" 被当作 0，其它情况若通过 IsValidNumber 检查则用 atof 转换，否则退回为 0。虚部部分中，空串或 "+" 被解释为系数 1，"-" 解释为 -1，其它非空字符串必须通过 IsValidNumber 检查后才能用 atof 解析；如果检查失败，整个输入被视为错误并设置 failbit。

整体上，这一解析逻辑在发现非法字符、'i' 位置异常或数值片段不合法时能及时返回错误状态，确保上层可以统一处理输入失败。

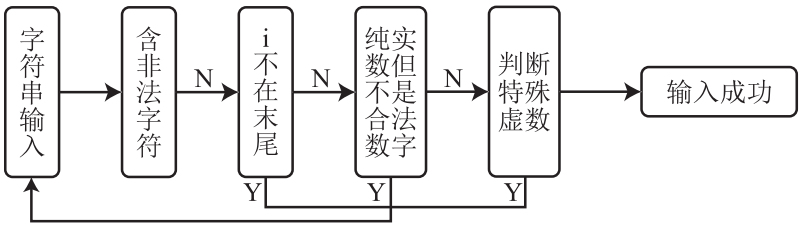

图 3 复数输入流程图

4. 复数输出格式设计——operator<<

输出运算符重载 ostream& operator<<(ostream& os, const Complex& A) 负责将内部存储形式转换为用户更易读的字符串形式。逻辑上按复数的值域分多种情况处理。

当 real $= 0$ 且 imag $= 0$ 时，直接输出 "0"，不再展示 i。当实部不为零时先输出实部；虚部不为零时根据符号与实部是否存在决定是否在前面附加 '+' 号：只有在已经输出了实部且虚部为正时才加号，否则直接接负号或数值本身。

虚部值的输出简化：当 imag $= 1$ 时输出 "i"；当 imag $= -1$ 时输出 "-i"；其它情况下输出数值再接字符 'i'，例如 3.5i 输出为 "3.5i"。通过这些分支，可以覆盖 $3+5$i、-2i、5、i 等常见表述形式。

5. 复数输入控制函数 Input 的设计

Input(Complex& A) 封装了对单个复数的交互式读入过程。函数内部使用一个无限循环，每次先输出提示信息，再尝试通过 cin >> A 调用前面定义的输入运算符完成解析。如果输入合法，operator>> 会成功写入 A 并保持流状态正常，Input 随即跳出循环；若输入格式不符合要求，operator>> 会设置 failbit，Input 检测到失败后输出错误提示，调用 cin.clear() 清除错误标志，使输入流可以继续接受下一次输入。

6. 菜单与运算封装函数的设计

菜单函数 menu(Complex& A, Complex& B) 每次被调用时都会显示当前的 A 和 B，输出格式由 operator<< 保证为类似 $a+b$i 的形式。紧接着给出编号为 0–5 的操作选项，包括修改复数、加减乘除运算以及退出。通过在每轮操作前重复展示当前状态，用户可以清楚地看到本轮运算所基于的两个复数。

加法、减法、乘法和除法分别由 Arr、Sub、Mul、Div 四个函数负责。前三个函数的结构一致：以 A 与 B 为输入，调用相应的运算符重载得到结果 C，随后按“(A) 运算符 (B) = C”的形式直接输出。例如加法输出 (A)+(B)=C，减法输出 (A)-(B)=C，乘法输出 (A)*(B)=C。其中 A、B、C 的显示由 operator<< 统一完成，输出为 $a+b$i 的格式。

Div 在进行 A/B 之前先调用 B.getModulus() 检查分母是否为零。如果模长为 0，说明 $B = 0+0$i，此时除法没有意义，直接输出“不能进行除法运算”并结束本次操作；否则调用除法运算符得到结果 C，再按与其他运算相同的格式输出算式与结果，并给出“除法运算完成”的提示。这样可以提前对非法运算作出局部拦截。

7. 主控流程 main 的设计

main 函数作为程序入口，负责串联各个模块并管理整轮交互逻辑。启动阶段先创建两个 Complex 对象 A 与 B，默认值为 $0+0$i，并在终端输出输入格式说明和示例，使用户了解可以使用的复数写法。随后依次调用两次 Input，获得初始的 A 和 B。

主循环通过 while(1) 实现。每轮循环先调用 menu(A,B) 展示当前状态和可选操作，再尝试从 cin 中读取一个整型变量 n 作为菜单选项。如果读入失败（例如用户输入了非数字字符），程序输出错误提示，调用 cin.clear() 清除错误状态，再用 cin.ignore(1024,'n') 丢弃本行剩余内容，随后调用 system("pause") 暂停，最后通过 continue 回到循环起点重新显示菜单，避免错误输入影响后续逻辑。

当成功读入整型选项后，switch 语句根据 n 的值调度不同功能：选项 1 通过两次 Input 重新输入 A 和 B；选项 2-5 分别对应 Arr、Sub、Mul、Div，执行相应运算并回到菜单；选项 0 输出退出提示，调用 system("pause") 保留终端窗口内容，然后通过 return 0 结束整个程序。对 0-5 以外的数字，程序输出错误提示，调用 system("pause") 暂停后重新回到菜单，等待用户输入新的选项。

三、程序清单及运行结果

1. 程序清单

Listing 1: student grades.cpp

```
#include<iostream>
#include<string>
#include<cmath>
#include<cstdlib>
using namespace std;

class Complex {
   private:
      double real;
      double imag;

   public:
      Complex(double r=0.0,double i=0.0) {
         real=r;
         imag=i;
      }
      double getModulus();
      Complex operator+(Complex &B);
      Complex operator-(Complex &B);
      Complex operator*(Complex &B);
      Complex operator/(Complex &B);
      friend ostream& operator<<(ostream& os, const Complex& A);
      friend istream& operator>>(istream& is, Complex& A);
};

double Complex::getModulus() {
   return (sqrt(real*real+imag*imag));
}

Complex Complex::operator+(Complex &B) {
   Complex C;
   C.real=real+B.real;
   C.imag=imag+B.imag;
   return C;
}
```

```
Complex Complex::operator-(Complex &B) {
   Complex C;
   C.real=real-B.real;
   C.imag=imag-B.imag;
   return C;
}

Complex Complex::operator*(Complex &B) {
   Complex C;
   C.real=real*B.real-imag*B.imag;
   C.imag=imag*B.real+real*B.imag;
   return C;
}

Complex Complex::operator/(Complex &B) {
   Complex C;
   double temp=B.real*B.real+B.imag*B.imag;
   C.real=(real*B.real+imag*B.imag)/temp;
   C.imag=(imag*B.real-real*B.imag)/temp;
   return C;
}

ostream& operator<<(ostream& os, const Complex& A) {
   if (A.real==0 && A.imag==0) {
      os<<"0";
      return os;
   }
   if (A.real!=0) {
      os<<A.real;
   }
   if (A.imag!=0) {
      if(A.real!=0 && A.imag>0)
         os<<"+";
      if(A.imag==1) os<<"i";
      else if(A.imag==-1)
         os<<"-i";
      else
         os<<A.imag<<"i";
   }
   return os;
}

bool IsValidNumber(string s) {
   if (s.empty()) return false;
   int dotCount = 0;
   for (int k=0; k < s.length(); k++) {
      if (k==0 && (s[k]=='-' || s[k]=='+'))
         continue;
      if (s[k]=='.') {
```

```
            dotCount++;
            if(dotCount>1)
               return false;
         } else if(s[k]<'0' || s[k]>'9') {
            return false;
         }
      }
      return true;
}

istream& operator>>(istream& is, Complex& A) {
      string strPtr;
      if(is.peek()=='\n')
         is.ignore();
      getline(is,strPtr);
      int n=strPtr.length();
      if (n==0) {
         is.setstate(ios::failbit);
         return is;
      }
      int i_pos=-1;
      for (int j=0; j<n; j++) {
         if(strPtr[j]=='i') {
            i_pos =j;
            break;
         }
         char c = strPtr[j];
         if (!((c>='0' && c<='9') || c=='.' || c=='+' || c=='-' || c=='i')) {
            is.setstate(ios::failbit);
            return is;
         }
      }
      if(i_pos!=-1 && i_pos!=n-1) {
         is.setstate(ios::failbit);
         return is;
      }
      if(i_pos==-1) {
         if(IsValidNumber(strPtr)) {
            A.real=atof(strPtr.c_str());
            A.imag=0;
         } else {
            is.setstate(ios::failbit);
         }
      }
      else {
         string temp=strPtr.substr(0, i_pos);
         int splitPos=-1;
         for(int k=temp.length()-1;k>0;k--) {
            if (temp[k]=='+' || temp[k]=='-') {
```

```
            splitPos=k;
            break;
        }
    }
    string s_real,s_imag;
    if(splitPos==-1) {
        s_real="0";
        s_imag=temp;
    } else {
        s_real=temp.substr(0, splitPos);
        s_imag=temp.substr(splitPos);
    }
    if(s_real=="" || s_real=="+")
        A.real=0;
    else if(IsValidNumber(s_real))
        A.real=atof(s_real.c_str());
    else
        A.real=0;
    if(s_imag=="" || s_imag=="+")
        A.imag=1;
    else if(s_imag=="-")
        A.imag=-1;
    else if(IsValidNumber(s_imag))
        A.imag=atof(s_imag.c_str());
    else {
        is.setstate(ios::failbit);
    }
  }
  return is;
}

void Input(Complex &A) {
  while(1) {
    cout<<"请输入一个虚数: ";
    if(cin>>A) {
        break;
    } else {
        cout<<"\n输入格式错误，请重新输入。\n";
        cin.clear();
    }
  }
}

void menu(Complex &A,Complex &B) {
  cout<<"\n当前A="<<A<<" B="<<B<<endl;
  cout<<"\n\n1. 修改复数\n";
  cout<<"2. 加法运算\n";
  cout<<"3. 减法运算\n";
  cout<<"4. 乘法运算\n";
```

```
    cout<<"5. 除法运算\n";
    cout<<"0. 退出系统\n\n";
}

void Arr(Complex &A,Complex &B) {
    Complex C;
    C=A+B;
    cout<<"("<<A<<")+("<<B<<")="<<C<<endl;
    cout<<"\n加法运算完成。\n";
}

void Sub(Complex &A,Complex &B) {
    Complex C;
    C=A-B;
    cout<<"("<<A<<")-("<<B<<")="<<C<<endl;
    cout<<"\n减法运算完成。\n";
}

void Mul(Complex &A,Complex &B) {
    Complex C;
    C=A*B;
    cout<<"("<<A<<")*("<<B<<")="<<C<<endl;
    cout<<"\n乘法运算完成。\n";
}

void Div(Complex &A,Complex &B) {
    Complex C;
    if(B.getModulus()==0)
        cout<<"该复数不能进行除法运算。\n" ;
    else {
        C=A/B;
        cout<<"("<<A<<")/("<<B<<")="<<C<<endl;
        cout<<"\n除法运算完成。\n";
    }
}

int main() {
    Complex A,B;
    cout<<"所有虚数必须按照a+bi的格式输入，";
    cout<<"如3+5i;-i;5。"<<endl;
    Input(A);
    Input(B);
    while(1) {
        int n;
        menu(A,B);
        if (!(cin>>n)) {
            cout <<"\n输入字符无效，请输入数字。\n";
            cin.clear();
            cin.ignore(1024,'\n');
```

```
            system("pause");
            continue;
        }
        switch(n) {
            case 1:
                Input(A);
                Input(B);
                cout<<"虚数修改完成。\n";
                break;
            case 2:
                Arr(A,B);
                break;
            case 3:
                Sub(A,B);
                break;
            case 4:
                Mul(A,B);
                break;
            case 5:
                Div(A,B);
                break;
            case 0:
                cout<<"\n正在退出中...\n";
                system("pause") ;
                return 0;
            default:
                cout<<"\n输入数字无效，请重新输入。\n";
                system("pause");
                break;
        }
    }
    system("pause") ;
    return 0;
}
```

2. 运行结果

```
所有虚数必须按照a+bi的格式输入，如3+5i;-i;5。
请输入一个虚数：0
请输入一个虚数：-12i

当前A=0 B=-12i



1．修改复数
2．加法运算
3．减法运算
4．乘法运算
```

```
5. 除法运算
0. 退出系统

1
请输入一个虚数：2-9i
请输入一个虚数：5-3i
虚数修改完成。

当前A=2-9i B=5-3i


1. 修改复数
2. 加法运算
3. 减法运算
4. 乘法运算
5. 除法运算
0. 退出系统

2
(2-9i)+(5-3i)=7-12i

加法运算完成。

当前A=2-9i B=5-3i


1. 修改复数
2. 加法运算
3. 减法运算
4. 乘法运算
5. 除法运算
0. 退出系统

3
(2-9i)-(5-3i)=-3-6i

减法运算完成。

当前A=2-9i B=5-3i


1. 修改复数
2. 加法运算
3. 减法运算
4. 乘法运算
5. 除法运算
0. 退出系统

4
```

```
(2-9i)*(5-3i)=-17-51i

乘法运算完成。

当前A=2-9i B=5-3i


1. 修改复数
2. 加法运算
3. 减法运算
4. 乘法运算
5. 除法运算
0. 退出系统

5
(2-9i)/(5-3i)=1.08824-1.14706i

除法运算完成。

当前A=2-9i B=5-3i


1. 修改复数
2. 加法运算
3. 减法运算
4. 乘法运算
5. 除法运算
0. 退出系统

0

正在退出中...
请按任意键继续. . .

--------------------------------
Process exited after 44.84 seconds with return value 0
请按任意键继续. . .
Process exited after 7.047 seconds with return value 0
```