Benjamin Czaczkes PhD

The Hebrew University of Jerusalem,
Programming Instruction Unit,
Givat Ram, Jerusalem 91904, Israel

Site:pl1.mscc.huji.ac.il :אתר e-mail: msbc@mscc.huji.ac.il :דואר אלקט

ד"ר בנימין צ'צ'קס

האוניברסיטה העברית בירושלים,
היחידה להוראת התכנות,
גבעת רם, ירושלים 91904

Fax: 972-2-6586246 :פקס Tel: 972-2-6586297 :טל'

תכנות פורטרן ו C++ פתרון מבחן Complex

מבחן בקורס תכנות C++ לתלמידי פיזיקה – 76641 מועד א' סמסטר ב' תשס"א

1. הגדירו מחלקה בשם complex המייצגת מספר מרוכב. במחלקה יהיו 2 משתנים, המוגנים לגישה מחוץ למחלקה, מסוג double המציינים חלק ממשי וחלק מדומה של מספר מרוכב.
2. הגדירו constructor למחלקה.
3. הגדירו את האופרטור + לחיבור בין מספר ממשי ומספר מרוכב.
4. הגדירו את האופרטור - לחיסור בין מספר ממשי ומספר מרוכב.
5. הגדירו את האופרטור / לחילוק בין מספר מרוכב ומספר ממשי.
6. כיתבו פונקציה למחלקה המחזירה את מרחק המספר המרוכב מראשית הצירים (יש להסתכל על מספר מרוכב כנקודה במרחב הדו ממדי).
7. כיתבו פונקציה חיצונית למחלקה המקבלת כפרמטרים את שלושת המקדמים של משוואה ריבועית (פרמטרים מסוג double). הפונקציה תחזיר, בכל מקרה, בשני פרמטרים את שני פתרונות המשוואה הריבועית גם במקרה שהם מרוכבים (פרמטרים מסוג complex).
8. נתון קובץ נתונים בשם cppdata.txt המכיל, בכל שורה, שלשה של מספרים ממשיים המהווים מקדמים של משוואה ריבועית. כיתבו תוכנית אשר קוראת קובץ זה ומדפיסה את השורש ה"קטן" ביותר מבין כל שורשי המשוואות הריבועיות. כלומר, אם יש n שורות בקובץ, יהיו 2n שורשים ויש למצוא את ה"קטן" ביותר מהם. מספר מרוכב x יוגדר כקטן מ y, אם מרחק x מראשית הצירים קטן ממרחק y מראשית הצירים. ניתן להניח כי קיים רק שורש אחד כזה.

הערות:

- הנכם יכולים להוסיף כל פונקציה נוספת שתמצאו לנכון.
- משקלה של כל שאלה מ 1 עד 6 הוא 8%. שאלה 7 12%. שאלה 8 40%.

בהצלחה

```
01  /************************************
02    File Name: c:\My Documents\cppf90\week11\complex\main.cpp
03    Revision By: Benjamin Czaczkes PhD
04    Revised on 15/06/02 16:38:14
05   ************************************/
06  #include <fstream.h>
07  #include <math.h>                                                    //for sqrt in q7
08  class complex                                                                 // q1
09  {
10  private:
11     double r,i;
12  public:
13     complex(void){ r=0.0; i=0.0; return; }                                     // q2
14     friend const complex operator+(const complex & lhs, const double & rhs);  // q3
15     friend const complex operator+(const double & lhs, const complex & rhs);  // q3
16     friend const complex operator-(const complex & lhs, const double & rhs);  // q4
17     friend const complex operator-(const double & lhs, const complex & rhs);  // q4
18     const complex operator/(const double & rhs);                              // q5
19     double q6(void)   { return sqrt(i*i+r*r); }                               // q6
20     void seti(double I) {i=I; return;}                                 //for q7
21     void setr(double R) {r=R; return;}                                 //for q7
22     const bool operator<(complex & rhs) { return q6()<rhs.q6(); }      //for q8
23     friend ostream & operator<<(ostream & lhs, const complex & rhs);   //for q8
24  };
25  void q7(double a, double b, double c, complex * r1, complex * r2);           //q7
26  void q8(void);                                                               //q8
27
28  void main(void)
29  {
30     complex r;
31     cout << r << endl ; // check q2
32     r=r+1.1;          // check q3
33     cout << r << endl ; // check q3
34     r.seti(1.0);            // check seti
35     r=1.1+r;          // check q3
36     cout << r << endl ; // check q3
37     r=r-1.1;          // check q4
38     cout << r << endl ; // check q4
39     r.setr(9.0);            // check setr
40     r=1.1-r;          // check q4
41     cout << r << endl ; // check q4
42     r=r/10.0;         // check q5
43     cout << r << endl ; // check q5
44     q8();
45     return;
46  }
47  const complex operator+(const complex & lhs, const double & rhs)   // q3
48  {
49     complex t;
50     t.i=lhs.i;
51     t.r=lhs.r+rhs;
52     return t;
53  }
54  const complex operator+(const double & lhs, const complex & rhs)   // q3
55  {
56     complex t;
57     t.i=rhs.i;
58      t.r=lhs+rhs.r;
59     return t;
60  }
61  const complex operator-(const complex & lhs, const double & rhs)   // q4
62  {
63     complex t;
64     t.i=lhs.i;
65     t.r=lhs.r-rhs;
66     return t;
67  }
68  const complex operator-(const double & lhs, const complex & rhs)   // q4
69  {
70     complex t;
71     t.i=rhs.i;
72     t.r=lhs-rhs.r;
73     return t;
74  }
```

```
75  const complex complex::operator/(const double & rhs)               // q5
76  {
77     complex t;
78     t.i=i/rhs;
79     t.r=r/rhs;
80     return t;
81  }
82  void q7(double a, double b, double c, complex * r1, complex * r2)  //q7
83  {
84     double d=b*b-4*a*c;
85     if (d<0.0)
86     {
87        r1->setr(-b/(2*a));
88        r2->setr(-b/(2*a));
89        r1->seti(-sqrt(4*a*c-b*b)/(2*a));
90        r2->seti(+sqrt(4*a*c-b*b)/(2*a));
91     }
92     else if (d>0.0)
93     {
94        double s=sqrt(d);
95        r1->setr((-b-s)/(2*c));
96        r2->setr((-b+s)/(2*c));
97        r1->seti(0.0);
98        r2->seti(0.0);
99     }
100    else
101    {
102       r1->setr(-b/(2*c));
103       r2->setr(-b/(2*c));
104       r1->seti(0.0);
105       r2->seti(0.0);
106    }
107    return;
108 }
109 void q8(void)                                        //q8
110 {
111    ifstream data;
112    data.open("cppdata.txt");
113    complex m,r1,r2;
114    double a, b, c;
115    if (data)
116    {
117       data >> a >> b >> c;
118       q7(a, b, c, &r1, &r2);
119       if (r1<r2) m=r1; else m=r2;
120    }
121    else
122    {
123       cout << "No Data in file" << endl;
124    }
125    while (data)
126    {
127       data >> a >> b >> c;
128       q7(a, b, c, &r1, &r2);
129       if (r1<m) m=r1;
130       if (r2<m) m=r2;
131    }
132    data.close();
133    cout << "min is:" << m << endl;
134    return;
135 }
136 ostream & operator<<(ostream & lhs, const complex & rhs)           //for q8
137 {
138    lhs << rhs.r << " " << rhs.i << "i";
139    return lhs;
140 }
```