

# **Welcome to C++**

**CSE 225 - Data Structures and Algorithms**

Md. Mahfuzur Rahman  
ECE Department  
North South University

## 1 frac.h

```
1 class FractionType
2 {
3 public:
4     void Initialize(int numerator, int denominator);
5     // Function: Initialize the fraction
6     // Pre: None
7     // Post: Fraction is initialized
8     int NumeratorIs();
9     // Function: Returns the value of the numerator
10    // Pre: Fraction has been initialized
11    // Post: numerator is returned
12    int DenominatorIs();
13    // Function: Returns the value of the denominator
14    // Pre: Fraction has been initialized
15    // Post: denominator is returned
16    bool IsZero();
17    // Function: Determines if fraction is zero
18    // Pre: Fraction has been initialized
19    // Post: Returns true if numerator is zero
20    bool IsNotProper();
21    // Function: Determines if fraction is a proper fraction
22    // Pre: Fraction has been initialized
23    // Post: Returns true if fraction is greater than one
24    int ConvertToProper();
25    // Function: Converts the fraction to a whole number and a
26    //           fractional part
27    // Pre: Fraction has been initialized, is in reduced form, and
28    //      is not a proper fraction
29    // Post: Returns whole number
30    //      Remaining fraction is original fraction minus the
31    //      whole number; fraction is in reduced form
32 private:
33     int num;
34     int denom;
35 };
```

## 2 frac.cpp

```
1 // Implementation file for class FractionType
2 #include "frac.h"
3 void FractionType::Initialize(int numerator, int denominator)
4 // Function: Initialize the fraction
5 // Pre: None
6 // Post: numerator is stored in num; denominator is stored in
7 //       denom
8 {
9     num = numerator;
10    denom = denominator;
11 }
12 int FractionType::NumeratorIs()
13 // Function: Returns the value of the numerator
14 {
15     return num;
16 }
17 int FractionType::DenominatorIs()
18 // Function: Returns the value of the denominator
19 {
20     return denom;
21 }
22
23 bool FractionType::IsZero()
24 // Function: Determines if fraction is zero
25 // Pre: Fraction has been initialized
26 // Post: Returns true if numerator is zero
27 {
28     return (num == 0);
29 }
30
31 bool FractionType::IsNotProper()
32 // Function: Determines if fraction is a proper fraction
33 // Pre: Fraction has been initialized
34 // Post: Returns true if num is greater than or equal to denom
35 {
36     return (num >= denom);
37 }
38
39 int FractionType::ConvertToProper()
40 // Function: Converts the fraction to a whole number and a
41 //           fractional part
42 // Pre: Fraction has been initialized, is in reduced form, and
43 //       is not a proper fraction
44 // Post: Returns num divided by denom
45 //       num is original num % denom; denom is not changed
46 {
47     int result;
48     result = num / denom;
49     num = num % denom;
50     return result;
51 }
```

### 3 fracDr.cpp

```
1 // Test driver
2 #include <iostream>
3 #include <fstream>
4 #include <string>
5 #include "frac.h"
6 int main()
7 {
8     using namespace std;
9     ifstream inFile;           // file containing operations
10    ofstream outFile;          // file containing output
11    string inFileName;          // input file external name
12    string outFileName;         // output file external name
13    string outputLabel;
14    string command;             // operation to be executed
15    int numCommands;
16    FractionType fraction;
17    // Prompt for file names, read file names, and prepare files
18    cout << "Enter name of input file; press return." << endl;
19    cin >> inFileName;
20    inFile.open(inFileName.c_str());
21
22    cout << "Enter name of output file; press return." << endl;
23    cin >> outFileName;
24    outFile.open(outFileName.c_str());
25
26    cout << "Enter name of test run; press return." << endl;
27    cin >> outputLabel;
28    outFile << outputLabel << endl;
29
30    inFile >> command;
31    numCommands = 0;
32    while (command != "Quit")
33    {
34        if (command == "Initialize")
35        {
36            int numerator, denominator;
37            inFile >> numerator;
38            inFile >> denominator;
39            fraction.Initialize(numerator, denominator);
40            outFile << "Numerator: " << fraction.NumeratorIs()
41                << " Denominator: " << fraction.DenominatorIs() << endl;
42        }
43        else if (command == "NumeratorIs")
44            outFile << "Numerator: " << fraction.NumeratorIs() << endl;
45        else if (command == "DenominatorIs")
46            outFile << "Denominator: " << fraction.DenominatorIs() << endl;
47        else if (command == "IsZero")
48            if (fraction.IsZero())
49                outFile << "Fraction is zero " << endl;
50        else
51            outFile << "Fraction is not zero " << endl;
```

```
52     else if (command == "IsNotProper")
53         if (fraction.IsNotProper())
54             outFile << "Fraction is improper " << endl;
55         else
56             outFile << "Fraction is proper " << endl;
57     else
58     {
59         outFile << "Whole number is " << fraction.ConvertToProper()
60             << endl;
61         outFile << "Numerator: " << fraction.NumeratorIs()
62             << " Denominator: " << fraction.DenominatorIs() << endl;
63     }
64 }
65
66 numCommands++;
67 cout << "Command number " << numCommands << " completed."
68     << endl;
69 inFile >> command;
70 };
71
72 cout << "Testing completed." << endl;
73 return 0;
74 }
```

## 4 fracIn

```
1 Initialize
2 3
3 4
4 IsZero
5 IsNotProper
6 NumeratorIs
7 DenominatorIs
8 Initialize
9 4
10 3
11 IsNotProper
12 ConvertToProper
13 Initialize
14 0
15 1
16 IsZero
17 Initialize
18 8
19 4
20 IsNotProper
21 ConvertToProper
22 Quit
```

## 5 fracOut

```
1 FinalRun
2 Numerator: 3 Denominator: 4
3 Fraction is not zero
4 Fraction is proper
5 Numerator: 3
6 Denominator: 4
7 Numerator: 4 Denominator: 3
8 Fraction is improper
9 Whole number is 1
10 Numerator: 1 Denominator: 3
11 Numerator: 0 Denominator: 1
12 Fraction is zero
13 Numerator: 8 Denominator: 4
14 Fraction is improper
15 Whole number is 2
16 Numerator: 0 Denominator: 4
```

## 6 DateType.h

```
1 #include <string>
2 #include <fstream>
3 using namespace std;
4 // Declare a class to represent the Date ADT
5 // This is file DateType.h.
6 enum RelationType {LESS, EQUAL, GREATER};
7 // Compares self with someDate.
8 class DateType
9 {
10 public:
11     void Initialize(int newMonth, int newDay, int newYear);
12     int GetMonth() const;           // returns year
13     int GetYear() const;           // returns month
14     int GetDay() const;            // returns day
15     string GetMonthAsString() const; // returns month as a string
16     DateType Adjust(int daysAway) const;
17     RelationType ComparedTo(DateType someDate) const;
18 private:
19     int year;
20     int month;
21     int day;
22 };
```



## 7 DateType.cpp

```
1 // File DateType.cpp contains the implementation of class DateType
2 #include "DateType.h"
3 #include <fstream>
4 #include <iostream>
5 using namespace std;
6
7 // Nmaes of days in each month
8 static int daysInMonth[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30,
9                             31, 30, 31};
10
11 // Nmaes of the months
12 static string conversionTable[] = {"Error", "January", "February",
13                                     "March", "April", "May", "June", "July", "August", "September",
14                                     "October", "November", "December"};
15
16 void DateType::Initialize(int newMonth, int newDay, int newYear)
17 // Post: If newMonth, newDay and newYear represent a valid date,
18 //       year is set to newYear;
19 //       month is set to newMonth;
20 //       day is set to newDay;
21 //       otherwise a string exception is thrown, stating the
22 //       first incorrect parameter.
23 {
24     if (newMonth < 1 || newMonth > 12)
25         throw string("Month is invalid");
26     else if (newDay < 1 || newDay > daysInMonth[newMonth])
27         throw string("Day is invalid");
28     else if (newYear < 1583)
29         throw string("Year is invalid");
30     year = newYear;
31     month = newMonth;
32     day = newDay;
33 }
34 int DateType::GetMonth() const
35 // Accessor function for data member month.
36 {
37     return month;
38 }
39
40 string DateType::GetMonthAsString() const
41 // Returns data member as a string
42 {
43     return conversionTable[month];
44 }
45
46 int DateType::GetYear() const
47 // Accessor function for data member year.
48 {
49     return year;
50 }
51
```

```
52 int DateType::GetDay() const // Accessor function for data member day.
53 {
54     return day;
55 }
56
57 RelationType DateType::ComparedTo(DateType aDate) const
58 // Post: Function value = LESS, if self comes before aDate.
59 //           = EQUAL, if self is the same as aDate.
60 //           = GREATER, if self comes after aDate.
61 {
62     if (year < aDate.year)
63         return LESS;
64     else if (year > aDate.year)
65         return GREATER;
66     else if (month < aDate.month)
67         return LESS;
68     else if (month > aDate.month)
69         return GREATER;
70     else if (day < aDate.day)
71         return LESS;
72     else if (day > aDate.day)
73         return GREATER;
74     else return EQUAL;
75 }
76
77 DateType DateType::Adjust(int daysAway) const
78 // Post: Function value = newDate daysAway from self
79 {
80     int newDay = day + daysAway;
81     int newMonth = month;
82     int newYear = year;
83     bool finished = false;
84     int daysInThisMonth;
85     DateType returnDate;
86     while (! finished)
87     {
88         daysInThisMonth = daysInMonth[newMonth];
89         if (newMonth == 2)
90             if (((newYear % 4 == 0) && !(newYear % 100 == 0))
91                 || (newYear % 400 == 0))
92                 daysInThisMonth++;
93         if (newDay <= daysInThisMonth)
94             finished = true;
95         else
96         {
97             newDay = newDay - daysInThisMonth;
98             newMonth = (newMonth % 12) + 1;
99             if (newMonth == 1)
100                 newYear++;
101         }
102     }
103     returnDate.Initialize(newMonth, newDay, newYear);
104     return returnDate;
105 }
```

## 8 DateDr.cpp

```
1  /*
2  *   Main.cpp
3  *   DateType
4  */
5
6  #include <fstream>
7  #include <string>
8  #include <iostream>
9  #include "DateType.h"
10 using namespace std;
11 int main()
12 {
13     string command;
14     int month, day, year;
15     DateType date, date2;
16     ofstream outFile;
17     int daysAway;
18
19     outFile.open("date.out");
20     cout << "Input a command or Quit to terminate the test" << endl;
21     cin >> command;
22
23     while (command != "Quit")
24     {
25         if (command == "Initialize")
26         {
27             cout << "Input a month, day, and year on one line" << endl;
28             cin >> month >> day >> year;
29             try
30             {
31                 date.Initialize(month, day, year);
32                 outFile << command << ": " << date.GetMonthAsString() << " " << date.
33                     GetDay() << ", " << date.GetYear() << endl;
34             }
35             catch(string msg)
36             {
37                 outFile << msg << endl;
38             }
39         }
40         else if (command == "GetMonth")
41             outFile << command << ": " << date.GetMonth() << endl;
42         else if (command == "GetMonthAsString")
43             outFile << command << ": " << date.GetMonthAsString() << endl;
44         else if (command == "GetDay")
45             outFile << command << ": " << date.GetDay();
46         else if (command == "GetYear")
47             outFile << command << ": " << date.GetYear();
48         else if (command == "ComparedTo")
49         {
50             cout << "Input a month, day, and year on one line" << endl;
51             cin >> month >> day >> year;
```

```

51     date2.Initialize(month, day, year);
52     outFile << command << endl;
53     switch (date.ComparedTo(date2))
54     {
55         case LESS : outFile << date.GetMonthAsString() << " "
56                     << date.GetDay() << ", " << date.GetYear();
57                     outFile << " comes before ";
58                     outFile << date2.GetMonthAsString() << " "
59                     << date2.GetDay() << ", " << date2.GetYear() << endl;
60                     break;
61         case GREATER: outFile << date2.GetMonthAsString() << " "
62                     << date2.GetDay() << ", "
63                     << date2.GetYear();
64                     outFile << " comes before ";
65                     outFile << date.GetMonthAsString() << " "
66                     << date.GetDay() << ", "
67                     << date.GetYear() << endl;
68                     break;
69
70         case EQUAL  : outFile << date.GetMonthAsString() << " "
71                     << date.GetDay() << ", "
72                     << date.GetYear();
73                     outFile << " and ";
74                     outFile << date2.GetMonthAsString() << " "
75                     << date2.GetDay() << ", "
76                     << date2.GetYear() << endl;
77                     outFile << " are equal " << endl;
78                     break;
79     }
80 }
81 else if ("DaysAway")
82 {
83     cout << "Input days away" << endl;
84     cin >> daysAway;
85     date2 = date.Adjust(daysAway);
86     outFile << command << endl;
87     outFile << date.GetMonthAsString() << " " << date.GetDay() << ", "
88     << date.GetYear();
89     outFile << " plus " << daysAway << " is ";
90     outFile << date2.GetMonthAsString() << " " << date2.GetDay() << ", "
91     << date2.GetYear() << endl;
92 }
93 else
94     cout << "Unrecognized command." << endl;
95 cout << "Input a command or Quit to terminate the test" << endl;
96 cin >> command;
97 }
98
99 outFile.close();
100 }

```