



The following sections provide some examples on classes. They explain also how to separate the program into `.h` and `.cpp` files:

1 The String class

```
1 // File: astring.h
2
3 #ifndef __STRING_CLASS
4 #define __STRING_CLASS
5
6 #include <iostream>
7 using namespace std;
8
9 class String
10 {
11 private:
12     char* str; // C-string containing the string appended by the null char
13     int n; // Number of characters in str, not including the null char
14 public:
15     String(); // Empty constructor
16     String(const char* cstr); // Constructor that takes CString
17     String(const String& s); // Copy constructor
18     ~String(); // Destructor
19     String operator + (const String& b) const; // Add String + String
20     String operator + (const char* b) const; // Add String + CString
21     String& operator += (const String& b);
22     String& operator = (const String& b); // Copy assignment
23     operator const char*() const; // Type conversion from String to CString
24     char& operator[](int i);
25 // The following functions are friends, not members:
26 friend String operator + (const char*, const String&);
27 friend istream& operator >> (istream&, String&);
28 friend ostream& operator << (ostream&, const String&);
29 };
30
31 #endif
```

```
1 // File: astring.cpp
2
3 #include "astring.h"
4 #include <cstring>
5 using namespace std;
6
7 String::String()
8 {
9     n=0;
10    str=0;
11 }
12
13 String::String(const char* cstr)
14 {
15     n=strlen(cstr);
16     str=new char[n+1];
17     strcpy(str,cstr);
18 }
19
20 String::String(const String& s)
21 {
22     n=s.n;
23     str=new char[n+1];
24     strcpy(str, s.str);
25 }
26
27 String::~~String()
28 {
29     if(str) delete[] str;
30 }
31
32 String String::operator + (const String& b) const
33 {
34     String r;
35     r.n=n+b.n;
36     if(r.n==0) return r;
37     r.str=new char[r.n+1];
38     if(str) strcpy(r.str, str);
39     if(b.str) strcpy(r.str+n, b.str);
40     return r;
41 }
```

```
1 String String::operator + (const char* b) const
2 {
3     String r;
4     int nb=strlen(b);
5     r.n=n+nb;
6     if(r.n==0) return r;
7     r.str=new char[r.n+1];
8     if(str) strcpy(r.str, str);
9     if(b) strcpy(r.str+n, b);
10    return r;
11 }
12
13 String& String::operator += (const String& b)
14 {
15     if(b.n==0) return *this;
16     int new_n=n+b.n;
17     char* new_str=new char[new_n+1];
18     if(str) strcpy(new_str, str);
19     if(b.str) strcpy(new_str+n, b.str);
20     n=new_n;
21     if(str) delete[] str;
22     str=new_str;
23     return *this;
24 }
25
26 String& String::operator = (const String& b)
27 {
28     n=b.n;
29     if(str) delete[] str;
30     str=0;
31     if(!b.str) return *this;
32     str=new char[n+1];
33     strcpy(str, b.str);
34     return *this;
35 }
36
37 String::operator const char*() const
38 {
39     return str;
40 }
```

```
1 char& String::operator[](int i)
2 {
3     return str[i];
4 }
5
6 istream& operator >> (istream& in, String& s)
7 {
8     char buf[200];
9     buf[0]=0;
10    in>>buf;
11    if(s.str) delete[] s.str;
12    s.n=strlen(buf);
13    s.str=0;
14    if(s.n==0) return in;
15    s.str=new char[s.n+1];
16    strcpy(s.str, buf);
17    return in;
18 }
19
20 ostream& operator << (ostream& out, const String& s)
21 {
22     out<<s.str;
23     return out;
24 }
25
26 String operator + (const char* a, const String& b)
27 {
28     String r;
29     int na=strlen(a);
30     r.n=na+b.n;
31     if(r.n==0) return r;
32     r.str=new char[r.n+1];
33     if(a) strcpy(r.str, a);
34     if(b.str) strcpy(r.str+na, b.str);
35     return r;
36 }
```

```
1 // File: main.cpp
2
3 #include "astring.h"
4 #include <iostream>
5 using namespace std;
6
7 int main()
8 {
9     String a, b, c;
10
11     cin>>a>>b;
12     cout<<a<<"-"<<b<<endl;
13
14     a="Hello"; b="-World";
15
16     b=a;
17     cout<<b<<endl; // Prints: Hello
18
19     a="Hello"; b="-World";
20     const char* pa="hello";
21     const char* pb="-world";
22
23     c=a+b; cout<<c<<endl; // Prints: Hello–World
24     c=pa+b; cout<<c<<endl; // Prints: hello–World
25     c=a+pb; cout<<c<<endl; // Prints: Hello–world
26     c=a+=b; cout<<a<<" "<<c<<endl; // Prints: Hello–World Hello–World
27
28     a = "Hello"; // Implicit conversion
29     b = (String)"World"; // Explicit conversion
30     c = static_cast<String>("Prog"); // Explicit conversion
31     cout<<a<<" "<<b<<" "<<c<<endl; // Prints: Hello World Prog
32
33     const char* x = a;
34     const char* y = (const char*)a;
35     const char* z = static_cast<const char*>(a);
36     cout<<x<<" "<<y<<" "<<z<<endl; // Prints: Hello Hello Hello
37
38     cout<<a[1]<<endl; // Prints: e
39     a[2]='x';
40     cout<<a<<endl; // Prints: Hexlo
41
42     return 0;
43 }
```

2 The Fraction class

```
1 // File: fraction.h
2
3 #ifndef __FRACTION_CLASS
4 #define __FRACTION_CLASS
5
6 #include <iostream>
7 using namespace std;
8
9 class Fraction
10 {
11 private:
12     int num; // numerator;
13     int den; // denominator;
14
15 public:
16     Fraction(int n=0, int d=1); // Constructor with default arguments
17     operator double(); // Type conversion from Fraction to double
18     Fraction operator + (const Fraction& b) const;
19     Fraction operator += (const Fraction& b);
20     Fraction& operator ++ (); // The prefix ++ operator
21     Fraction operator ++ (int); // The postfix ++ operator
22
23 friend ostream& operator << (ostream& out, const Fraction& f);
24 };
25
26 #endif
```

```
1 // File: fraction.cpp
2
3 #include "fraction.h"
4
5 Fraction::Fraction(int n, int d)
6 {
7     if(d==0) d=1; // Avoid division by zero
8     this->num = n; this->den = d;
9 }
10
11 Fraction::operator double()
12 {
13     return (double) this->num / this->den;
14 }
15
16 Fraction Fraction::operator + (const Fraction& b) const
17 {
18     Fraction c(num * b.den + b.num * den, den * b.den);
19     return c;
20 }
21
22 Fraction Fraction::operator += (const Fraction& b)
23 {
24     *this = *this + b; // Use the overloaded + operator!
25     return *this;
26 }
27
28 Fraction& Fraction::operator ++ ()
29 {
30     num += den;
31     return *this;
32 }
33
34 Fraction Fraction::operator ++ (int)
35 {
36     Fraction f = *this;
37     num += den;
38     return f;
39 }
40
41 ostream& operator << (ostream& out, const Fraction& f)
42 {
43     out << f.num << "/" << f.den; return out;
44 }
```

```
1 // File: main.cpp
2
3 #include "fraction.h"
4 #include <iostream>
5 using namespace std;
6
7 int main()
8 {
9     Fraction a, b(2,3), d;
10
11     Fraction c=Fraction(7,4); // The RHS constructs a temporary object of type
12                               // Fraction using the two arguments constructor, then uses
13                               // the copy constructor to copy the temporary object into
14                               // the Fraction object 'c' in the LHS. But, since the temporary
15                               // object will not be used any more, the compiler usually
16                               // optimizes this line to be just the same as: Fraction c(7,4);
17                               // to avoid calling the copy constructor
18
19     cout<<a<<" "<<b<<" "<<c<<endl; // Prints: 0/1 2/3 7/4
20
21     d=b+c; cout<<d<<endl; // Prints: 29/12
22
23     cout << (double)d << endl; // Prints: 2.42
24
25     a=Fraction(2,3); // The RHS constructs a temporary object of type Fraction
26                     // using the two arguments constructor, then uses the
27                     // assignment operator to copy the temporary object into
28                     // the Fraction object 'a' in the LHS
29
30     b=Fraction(3,5); // Construct a temporary object then assigns it to 'b'
31
32     a+=b; cout<<a<<" "<<b<<endl; // Prints: 19/15 3/5
33
34     a=Fraction(3,5);
35
36     b=++a; cout<<a<<" "<<b<<endl; // Prints: 8/5 8/5
37
38     c=a++; cout<<a<<" "<<c<<endl; // Prints: 13/5 8/5
39
40     return 0;
41 }
```