Cairo University
Faculty of Computers and Information
Computer Science Department

Programming-2 CS213
2018/2019

Example classes

Dr. Amin Allam

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    delete[] str;
30 }
31
32 String String::operator + (const String& b) const
33 {
34    String r;
35    r.str=new char[n+b.n+1];
36    strcpy(r.str, str);
37    strcpy(r.str+n, b.str);
38    r.n=n+b.n;
39    return r;
40 }
```

```
1  String String::operator + (const char* b) const
2  {
3     String r;
4     int nb=strlen(b);
5     r.str=new char[n+nb+1];
6     strcpy(r.str, str);
7     strcpy(r.str+n, b);
8      r.n=n+nb;
9     return r;
10 }
11
12 String& String::operator += (const String& b)
13 {
14    int new_n=n+b.n;
15    char* new_str=new char[new_n+1];
16    strcpy(new_str, str);
17    strcpy(new_str+n, b.str);
18    n=new_n;
19    delete[] str;
20    str=new_str;
21    return *this;
22 }
23
24 String& String::operator = (const String& b)
25 {
26    delete[] str;
27    n=b.n;
28    str=new char[n+1];
29    strcpy(str, b.str);
30    return *this;
31 }
32
33 String::operator const char*() const
34 {
35    return str;
36 }
37
38 char& String::operator[](int i)
39 {
40    return str[i];
41 }
```

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

```cpp
// File : main.cpp

#include "astring.h"
#include <iostream>
using namespace std;

int main()
{
   String a, b, c;

   cin>>a>>b;
   cout<<a<<"-"<<b<<endl;

   a="Hello"; b="-World";

   b=a;
   cout<<b<<endl;   // Prints: Hello

   a="Hello"; b="-World";
   const char* pa="hello";
   const char* pb="-world";

   c=a+b;  cout<<c<<endl;   // Prints: Hello-World
   c=pa+b;  cout<<c<<endl;   // Prints: hello-World
   c=a+pb;  cout<<c<<endl;   // Prints: Hello-world
   c=a+=b;  cout<<a<<" "<<c<<endl;   // Prints: Hello-World Hello-World

   a = "Hello";   // Implicit conversion
   b = (String)"World";   // Explicit conversion
   c = static_cast<String>("Prog");   // Explicit conversion
   cout<<a<<" "<<b<<" "<<c<<endl; // Prints: Hello World Prog

   const char* x = a;
   const char* y = (const char*)a;
   const char* z = static_cast<const char*>(a);
   cout<<x<<" "<<y<<" "<<z<<endl; // Prints: Hello Hello Hello

   cout<<a[1]<<endl;   // Prints: e
   a[2]='x';
   cout<<a<<endl;   // Prints: Hexlo

   return 0;
}
```

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
```

```cpp
// File : fraction .cpp

#include "fraction.h"

Fraction::Fraction(int n, int d)
{
   if(d==0) d=1; // Avoid division by zero
   this->num = n; this->den = d;
}

Fraction::operator double()
{
   return (double) this->num / this->den;
}

Fraction Fraction::operator + (const Fraction& b) const
{
   Fraction c(num * b.den + b.num * den, den * b.den);
   return c;
}

Fraction Fraction::operator += (const Fraction& b)
{
   *this = *this + b;    // Use the overloaded + operator!
   return *this;
}

Fraction& Fraction::operator++()
{
   num += den;
   return *this;
}

Fraction Fraction::operator++(int)
{
   Fraction f = *this;
   num += den;
   return f;
}

ostream& operator << (ostream& out, const Fraction& f)
{
   out << f.num << "/" << f.den; return out;
}
```

```cpp
// File : main.cpp

#include "fraction.h"
#include <iostream>
using namespace std;

int main()
{
   Fraction a, b(2,3), c(7,4), d;

   cout<<a<<" "<<b<<" "<<c<<endl; // Prints: 0/1 2/3 7/4

   d=b+c; cout<<d<<endl; // Prints: 29/12
   cout << (double)d << endl; // Prints: 2.42

   a=Fraction(2,3); b=Fraction(3,5);
   a+=b; cout<<a<<" "<<b<<endl;  // Prints: 19/15 3/5

   a=Fraction(3,5);
   b=++a; cout<<a<<" "<<b<<endl;  // Prints: 8/5 8/5
   c=a++; cout<<a<<" "<<c<<endl;  // Prints: 13/5 8/5

   return 0;
}
```