CS 101: Computer Programming and Utilization

Shivaram Kalyanakrishnan
(Abhiram Ranade's slides, borrowed and edited)
Lecture 24

Today’s Lecture

- Structures: initialisation lists
- Operators and overloading
- Access control
- Classes
- Classes for I/O, file handling

Operator overloading

- In Mathematics, arithmetic operators are used with numbers, but also other objects such as vectors.

- Something like this is also possible in C++!

Consider `x @ y` where @ is any “infix” operator .

- C++ treats this as: `x.operator@(y)`

- `operator@` **must be a member function.**

- If the member function `operator@` is defined, then that is called to execute `x @ y`.

Example: arithmetic on V3 objects

```
int main(){
  V3 p(1,2,3),
q(4,5,6);
  V3 r, s;
  r = p+q;
  // r = p.operator+
(q);
  s = r * 10;
  // s =
r.operator@(10);
}
```

```
struct V3{
 double x, y, z;
 V3(double a, double b, double c){
    x=a; y=b; z=c;
 }
 V3(){}
 V3 operator+(V3 b){// V3 + V3
  return V3(x+b.x, y+b.y, z+b.z);
  // constructor call
 }
 V3 operator*(double f){ // V3 *
number
  return V3(x*f, y*f, z*f);
 }
};
```

Using V3 arithmetic

```
int main(){
  V3 u(1,2,3), a(4,5,6),
     s(0,0,0);
  double t=10;
  s = u*t + a*t*t*0.5;
  cout << s.x <<‘ ‘<< s.y <<‘ ‘
       << s.z << endl;
}
```

Today’s Lecture

- Structures: initialisation lists
- Operators and overloading
- Access control
- Classes
- Classes for I/O, file handling

Access Control

- It is possible to restrict access to members or member functions of a struct.
- Members declared `public`: no restriction.
- Members declared `private`: Can be accessed only inside the definition of the struct.
- Typical strategy: Declare all data members to be `private`, and some subset of function members to be `public`.

Access control example

```
struct Queue{
private:
  int elements[N], nWaiting, front;
public:
  Queue(){ … }
  bool insert(int v){
    ..
  }
  bool remove(int &v){
    ..
  }
};
```

Remarks

- `public:`, `private:` : access specifiers.

- An access specifier applies to all members defined following it, until another specifier is given.

- Thus `elements`, `nWaiting`, `front` are private, while `Queue()`, `insert`, `remove` are public.

- You can decide how structs work with operators

- Thus, as a designer of a struct, you can exercise great control over how the struct gets used.

Today’s Lecture

- Structures: initialisation lists
- Operators and overloading
- Access control
- Classes
- Classes for I/O, file handling

Classes

- A class is essentially the same as a struct, except:
  - Any members/member functions in a struct are public by default.
  - Any members/member functions in a class are private by default.
- Example: a Queue class:

```
class Queue{
  int elements[N], nWaiting, front;
public:
  Queue(){…}
  bool remove(int &v){…}
  bool insert(int v){…}
};
```

- Members `elements, nWaiting` and `front` will be private.

Input output classes

- `cin, cout` : objects of class `istream`, `ostream` resp.
  - Need to include header file <iostream>
  - got included because you included <simplecpp>
- `<<, >>` : operators defined for the objects of these classes.
- `ifstream`: another class like `istream`.
- You create an object of class `ifstream` and associate it with a file on your computer.
- Now you can read from that file by invoking the `>>` operator!
- `ofstream`: a class like `ostream`, to be used for writing to files.
  - Must include header file `<fstream>` to uses `ifstream` and `ofstream`.

Example of file i/o

```
#include <fstream>
#include <simplecpp>
int main(){
  ifstream
infile(“f1.txt”);.
  ofstream
outfile(“f2.txt”);
  repeat(10){
    int v;
    infile >> v;
    outfile << v<<endl;
  }
}
```

- Constructor call. object `infile` is created and associated with `f1.txt`, which must be present in the current directory.
- Constructor call. Object `outfile` is created and associated with `f2.txt`, which will get created in the current directory.
- Input and output can be performed using familiar >> and <<
- `f1.txt` must begin with 10 numbers. These will be read and written to file `f2.txt`.

Today’s Lecture

- Structures: initialisation lists
- Operators and overloading
- Access control
- Classes
- Classes for I/O, file handling