

cisc1110 fall 2010 lecture IX.1

- simple classes

simple classes

- classes are ways of organizing programs to provide structure
- a class is a special kind of *compound* data type
- classes are compound because they have *members*
- there are two types of members in classes:
 - *data* members
 - *function* members
- here is the syntax for defining a class:

```
class <class-name> {  
    public:  
        // declare data members  
        <class-member-data-type> <class-data-member-name>;  
        <class-member-data-type> <class-data-member-name>;  
        // declare function members  
        <class-member-data-type> <class-function-member-name> ( <arguments> ) {  
            <body of function>  
        } // end of function  
}; // end of class definition
```

where you fill in your own values for everything in angle brackets <...>

- things to notice:
 - two new C++ keywords: `class` and `public`
 - there is a semi-colon at the END OF THE CLASS DEFINITION, after the last curly brace `};`

- example definition:

```
class game_data {  
    public:  
        // declare data members  
        int num2;  
        string word;  
        int num3;  
        double num1;  
}; // end of game_data class definition
```

where `game_data` is the name of the class
the class contains four data members:
`num2`, `word`, `num3` and `num1`

- we can expand the example to define a function member for the class:

```
class game_data {  
    public:  
        // declare data members  
        int num2;  
        string word;  
        int num3;  
        double num1;  
        // declare function member  
        void print() {  
            cout << "word = [" << word << "]\n";  
            cout << "num1 = " << num1 << endl;  
            cout << "num2 = " << num2 << endl;  
            cout << "num3 = " << num3 << endl;  
        } // end of print()  
}; // end of game_data class definition
```

- we use classes by declaring variables whose data type is a class, defined using the name of that class; e.g.: `game_data x`;
- we call `x` an *object* of type `game_data`
- the *dot operator* (`.`) is used to indicate the member of a class
- then we can use the `game_data` member functions to operate on the object `x`, e.g.:

```
game_data x;
x.num1 = 0.53;
x.num2 = 12;
x.num3 = 45;
x.print();
```

notice the `x.` ("x dot") notation

complete example

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;

class game_data {
public:
    int num2;
    string word;
    int num3;
    double num1;
    // declare a function that prints the contents of the class
    void print() {
        cout << "word = [" << word << "]\n";
        cout << "num1 = " << num1 << endl;
        cout << "num2 = " << num2 << endl;
        cout << "num3 = " << num3 << endl;
    }
};
```

```
} // end of print()
}; // end of game_data class definition
```

```
int main() {
    ifstream infile;
    infile.open( "g2.dat" );
    if ( ! infile ) {
        cout << "error opening file :-(\n";
        //exit( 1 );
        return( 1 ); // only in the main()
    }
}
```

```
// declare variable: mydata object
game_data mydata;
```

```
// read data in input file
infile >> mydata.word;
infile >> mydata.num1;
infile >> mydata.num2;
```

```
infile >> mydata.num3;

// display data from input file
mydata.print();

// close file
infile.close();
} // end of main()
```