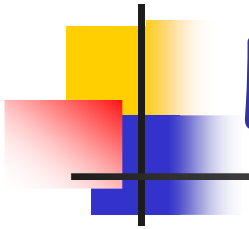
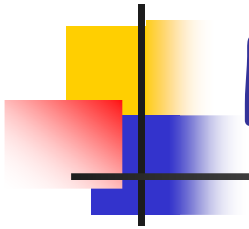
COMS W3101-1 Programming Language: C++ (Fall 2007)

Ramana Isukapalli



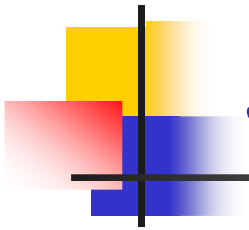
Lecture-1

- Course overview
  - See http://www.cs.columbia.edu/~ramana
- Overview of C
- Introduction to C++



Prerequisites

- A good knowledge of C programming is recommended.
- A good background in at least one programming language is required.

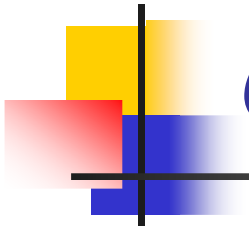


Syllabus Overview

- Overview of C
  - We will NOT cover details of C programming
- Object Oriented Programming principles wrt C++
  - Concepts of class/object, methods, inheritance, polymorphism, abstraction, data encapsulation

Overview of C programming language

- Basic data types
  - char, short, int, long, long long, unsigned, float, double, long double, …
- Operators:
  - Arithmetic: +, -, *, /, %, ++, --
  - Logical: ==, !=, >, <, >=, <=, &&, ||, !
  - Bitwise: &, |, ^, <<, >>, ~
- Complex data types
  - Struct
- Input, output
- Control statements
  - if else
  - for
  - while
  - switch, case

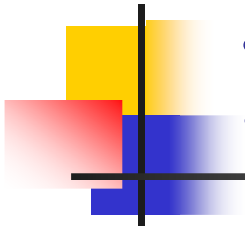


C structs

- C struct
  - used to contain > 1 basic data types
  - Can contain other structs
- E.g.,

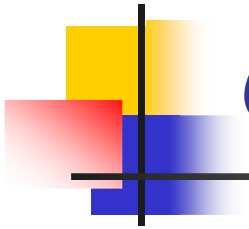
```
typedef struct
{
   int a, b, c;
   float x,y,z;
} myStruct;

myStruct m;
m.a = 1;
```



Input, Output

- Input
  - scanf – read input from std. input
    - E.g. scanf ("%d %s", &i, str);
    - Reads the values of i and str from std. Input
  - Others
    - fscanf, read, fread – we will not use in this class
- Output
  - printf – print output to std. Output
    - E.g. printf ("%d %s", i, str);
    - Print values of i and str to std. Output
  - Others
    - fprintf, write, fwrite – we will not use in this class



Control statements … if

```
if (<expr_1>)
{
   <body of if_expr_1>
}
else if(<expr_2>)
{
   < body of if_exp_2>
}
…
else /* default */
{
   …
}
```

- **Example-1**

```
if (i > j)
   printf ("i is larger\n");
```

- **Example-2**

```
if (i > j)
   printf ("i is larger\n");
else
   printf ("j is larger\n");
```

- **Example-3**

```
if (i > j)
   …
Else if (i > k)
   …
Else
 …
```

Control statements - for

- For (<start_expr>;
  <termination_cond>;
  <loop_increment>)

{

  <body_of_for>

}

- Example-1 /* print 0 to 9 */

```
for (i = 0; i < 10; i++)
{
   printf ("%d: \n", i);
}
```

- Example-2

```
For ( ;  ;)  /* infinite loop */
{
   /* do something */
}
```

Control statements – while

- Similar to for statement
- while (<while_cond>)

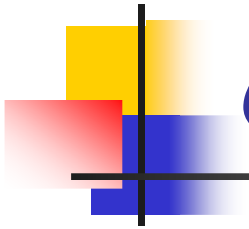
```
{
        <while_body>
}
```

- Example-1 /* print 0 to 9 */

```
while (i < 10)
{
        printf ("%d\n", i);
}
```

- Example-2

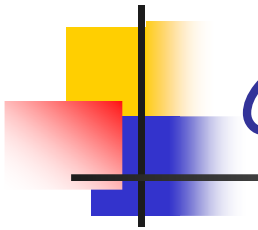
```
while (true) /* infinite loop */
{
        /* do something */
}
```



Control Statements – switch, case

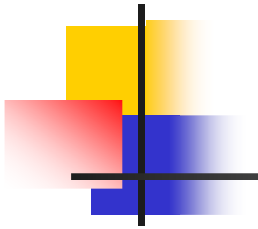
```
switch (x)
{
    case val1:
        <val1_body>;
        break;
    case val2:
        <val2_body>;
        break;
    …
    default:
        <default_body>
}
```

```
int x = 2;
switch (x)
{
    case 1:
        procedure1();
        break;
    case 2:
        procedure2(); /* executed*/
        break;
    …
    default:
        default_procedure();
}
```



C++ — Philosophically different from C

- High level features of C++
  - Uses concepts of "object oriented programming" (OOP)
  - Everything that works in C works in C++
    - C syntax, operators, structures, control statements, etc. work in C++
    - Reverse is NOT true
- Object Oriented Programming
  - Concept of class/object, methods, inheritance, encapsulation, abstraction, polymorphism
  - Key concepts in this
    - Separation of data and methods

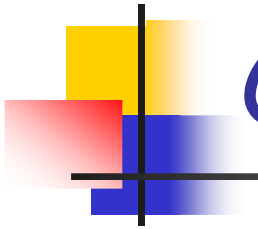


A simple "account" example

```
class account
{
      int user_SSN;                  // attribute (data)
      int accountNumber;             // attribute (data)
      void withdrawMoney (int amount);  // method
      void depositMoney (int amount);   // method
};
account x;  // x is an object of class "account"
```

Note:

1. class has both "attributes" and "methods".
2. Attributes and methods are "members" of a class
3. An instance of a class is an object.
4. A class should typically correspond to some meaningful entity.
5. A class uses methods to interact with other classes/functions.



Class methods

Syntax:

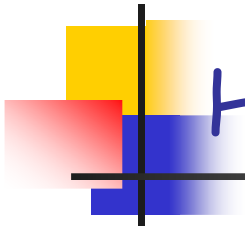
```
<ret_type> class::functionName(args)
{
   // code
}
```

Method code can be present in class definition

- Outside the class definition
- In a separate file

Example

```
Void account::withdrawMoney (int amount)
{
   // code
}
```



How do we initialize and cleanup objects?

```
class account
{
    int user_SSN;
    int accountNumber;
    account( );     // constructor – used to initialize account objects
    account(int ssn, int acctNum); // constructor
    ~account( );   // destructor – used to cleanup resources
    void withdrawMoney (int amount);
    void depositMoney (int amount);
};
```

Constructor

- A function with the same name as the class
- Called when an object is created
- A class can have more than one constructor

Destructor

- Called when an object is cleaned up (goes out of scope)
- One class can have only one destructor

Constructor and destructor

Constructor code

```
account::account( )
{ user_ssn = -1; accountNumber = -1; }

account::account( ) : user_ssn (-1), accountNumber(-1) { }

account::account (int ssn, int acctNum)
{
      user_ssn = ssn;
      accountNumber = acctNum;
}
```

Destructor code

```
~account::account( )
{  // Any memory/resource cleanup, etc. }
```

Examples

```
account x; // constructor code is called
account *y = new account(); // constructor code is called
delete (y); // destructor code is called
```