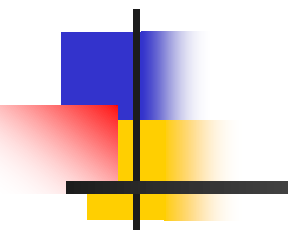
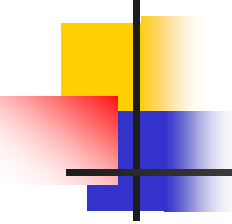


# COMS W3101 Programming Language: C++ (Spring 2016)



---

Ramana Isukapalli  
[ramana@cs.columbia.edu](mailto:ramana@cs.columbia.edu)



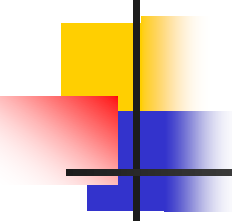
# Lecture-1

---

- Course overview

- See

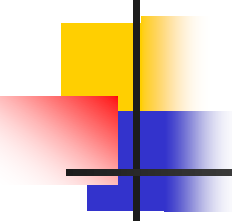
- <http://www.cs.columbia.edu/~ramana>



# Prerequisites

---

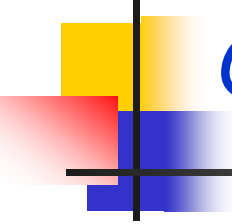
- A good background in at least one programming language is **recommended**.
- Or, ability to learn programming "quickly" - in about a week.



# 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



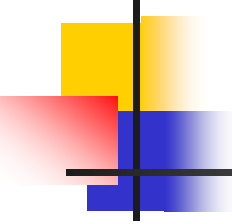
# Overview of C programming language

- Basic data types

- char, short, int, long, long long, unsigned, float, double, long double, ...

- Operators:

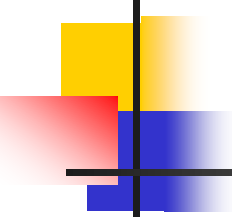
- Arithmetic: +, -, \*, /, %, ++, --
- Logical: ==, !=, >, <, >=, <=, &&, ||, !
- Bitwise: &, |, ^, <<, >>, ~



# Overview of C, contd.

---

- Input, output
- Control statements
  - if else
  - for
  - while
  - switch, case



# Control statements ... if

```
if (<expr_1>)
{
    <body of if_expr_1>
}
else if(<expr_2>)
{
    < body of if_exp_2>
}

else /* default */
{
}

}
```

## ■ Example-1

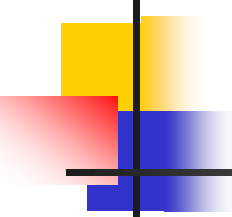
```
if (i > j)
    printf ("i is larger\n");
```

## ■ Example-2

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

## ■ Example-3

```
if (i > j)
{ }
else if (i > k)
{ }
else { }
```

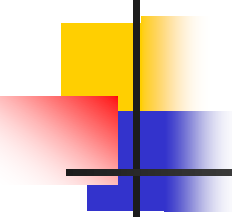


# Control statements - for

- for (<start\_expr>;  
    <termination\_cond>;  
    <loop\_increment>)  
{  
    <body\_of\_for>  
}

- Example-1 */\* print 0 to 9 \*/*  
for (i = 0; i < 10; i++)  
{  
    cout << i << endl;  
}  
■ Example-2  
for ( ; ; ) */\* infinite loop \*/*  
{  
    */\* do something \*/*  
}





# Control statements - while

- Similar to for statement

- `while ( <while_cond> )`  
`{`  
    `<while_body>`  
`}`

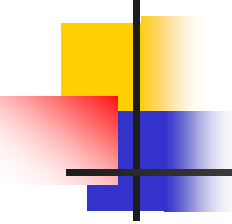
`do`  
`{`  
    `<body_of_do>`  
`} while (condition);`

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

```
i = 0;
while (i < 10)
{
    cout << i << endl;
    i++;
}
```

- Example-2

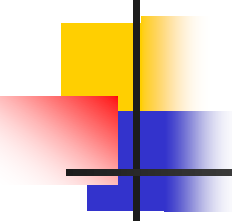
```
while (1) /* infinite loop */
{
    /* do something */
}
```



# Control Statements - switch, case

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

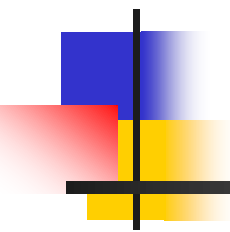
```
int x = 2;
switch (x)
{
    case 1:
        procedure1();
        break;
    case 2:
        procedure2(); /* executed */
        break;
    ...
    default:
        default_procedure( );
}
```



# Data types, IO, control statements

---

- C data types, IO and control statements work in C++
- C++ defines additional IO.
- Popular among that
  - cout
  - cin
- Advantage of cout and cin over printf, scanf
  - No need for %d, %s, %c, etc



# Arrays

---

# Arrays

- Arrays - an ordered sequence of elements of the same type.

- One dimensional array

|   |   |   |   |    |
|---|---|---|---|----|
| 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|----|

- `arr1[0] = 2; arr1[1] = 4 ...`

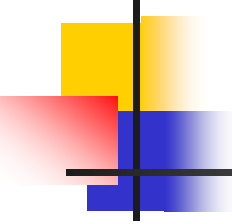
- Two dimensional array

|    |    |    |
|----|----|----|
| 5  | 10 | 15 |
| 20 | 25 | 30 |

- E.g.-2: `arr2`

- `arr2[0][0] = 5; arr2[0][1] = 10; arr2[0][2] = 15;`

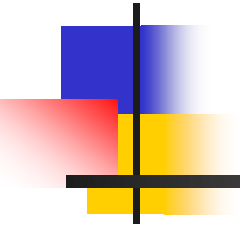
`arr2[1][0] = 20; arr2[1][1] = 25; arr2[1][2] = 30;`



# Arrays ... contd.

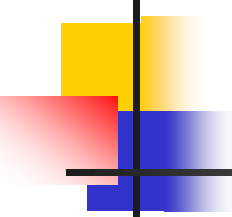
---

- Array of ints
  - `int intArray1[ ] = {2, 4, 6, 8, 10};`
- Array of floats
  - `float floatArray1[ ] = {1.1, 2.2, 3.3};`
- character array
  - `char str[] = "abcdef";`



C - character arrays

---



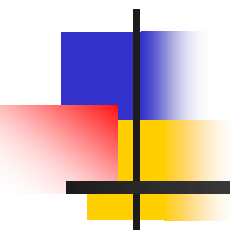
# C uses character arrays for strings

- C uses character arrays for strings.

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| C | O | L | U | M | B | I | A |
|---|---|---|---|---|---|---|---|

- Useful string functions
  - strlen - find the length of a string.
  - strcmp - compares two strings
    - Returns 0 if they match.
  - strstr - check if a string is sub-string of another string.
  - strcat - concatenate two strings.
  - Many others





# C++ string class

---



# C++ strings

- C uses char arrays to represent strings
- char arrays are messy
  - Need to predefine the size of array
  - Size can't be increased easily for longer strings.
  - Copying strings need to use strcpy.
- C++ strings - don't have these issues.
  - E.g. `string str1 = "abc";`  
`string str2 = str1;`  
`string str3 = str1 + "pqr";`  
Much more convenient than C character arrays