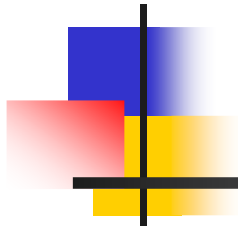
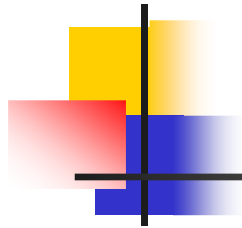


COMS W3101 Programming Language: C++ (Fall 2012)

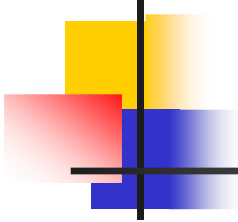


Ramana Isukapalli
ramana@cs.columbia.edu



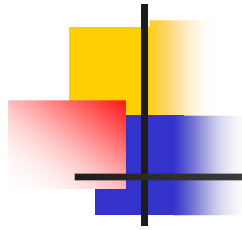
Lecture-1

- Course overview
 - See <http://www.cs.columbia.edu/~ramana>
- Overview of C
 - C data types
 - Input and output
 - Control statements - *if, for, while, do*
 - Functions
 - Pointers



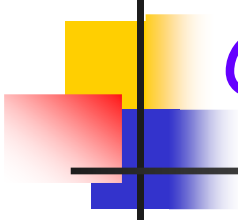
Prerequisites

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



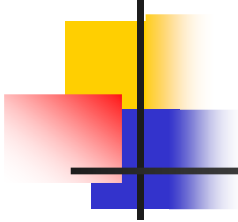
Syllabus Overview

- 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



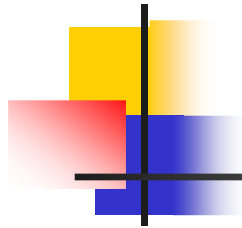
Overview of C programming language

- Basic data types
 - char, short, int, long, long long, unsigned, float, double, long double, ...
- Operators:
 - Arithmetic: +, -, *, /, %, ++, --
 - Logical: ==, !=, >, <, >=, <=, &&, ||, !
 - Bitwise: &, |, ^, <<, >>, ~
- Complex data types
 - struct



Overview of C, contd.

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

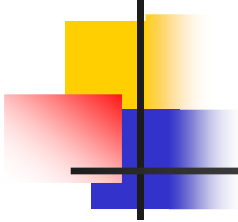


C structs

- 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;
```



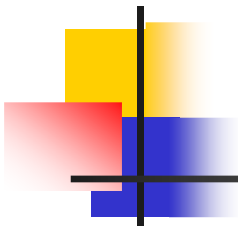
Input, Output

- 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`

- 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`



Control statements ... if

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

■ Example-1

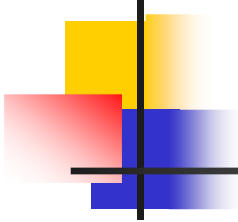
```
if (i > j)
    printf ("i is larger\n");
```

■ Example-2

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

■ Example-3

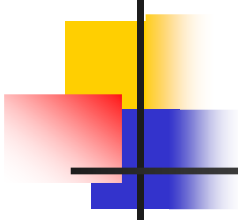
```
if (i > j)
    ...
else if (i > k)
    ...
else
```



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>
`}`

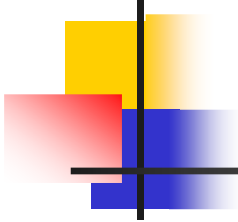
`do`
`{`
 <body_of_do>
`} while (condition);`

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

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

- Example-2

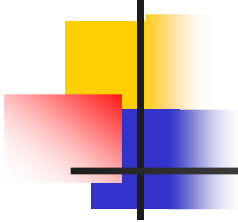
```
while (1) /* infinite loop */
{
    /* do something */
}
```



Control Statements - switch, case

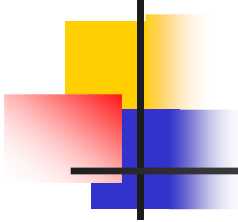
```
switch (x)
{
    case val1:
        <val1_body>;
        break;
    case val2:
        <val2_body>;
        break;
    ...
    default:
        <default_body>
}
```

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



C pointers

- A pointer “points” to a memory location.
 - E.g., `int x; /* x is an integer */`
`int *y; /* y points to an integer */`
`x = 5;`
`*y = 6; /* NOT y = 6 ! */`
- Pointers can point to any data type;
 - `int *x;      short *y;      char *str;`
 - `double *z; void *p, etc.`



C pointers, contd.

- Why do we need pointers?
 - Mainly to manage the memory, as opposed to the compiler managing memory.
 - User needs to assign and delete memory.
 - Allocate memory using `malloc`.
 - Delete memory using `free`.
- Examples
 - ```
int *x = (int *) malloc (sizeof (int));
*x = 3;
free (x);
```



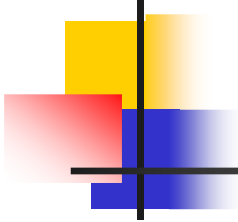
# C functions

---

- A group of statements
  - To perform a task
  - Possibly return a value

- Syntax

```
<return_type> fn_name (arguments)
{
 // function body
}
```



# C functions ... example

---

- Example

```
int square (int x) /* fn to compute square */
{
 return (x * x);
}

void main() /* Starting point of ANY C program */
{
 int i = 5;
 int i_sq = square (5);
 printf ("Square of 5 is: %d\n", i_sq);
}
```