

Programming in C

Part – 1: Introduction

Resources:

1. Stanford CS Education Library URL:
<http://cslibrary.stanford.edu/101/>
2. Programming in ANSI C, E Balaguruswamy,
Tata McGraw-Hill

PROGRAMMING IN C

A computer program is a collection of instructions for performing some specified task(s) with the computer.

- C is the offspring of **Basic Combined Programming Language** ('B') developed in 1960s by Cambridge University
- 'B' was Modified by Dennis Ritchie at Bell Laboratories in 1972, and was named 'C'.
- **Pros and cons:**
 - Programmers language. Gets in programmers way as little possible.
 - You need to know exactly what you are doing !! Can be Irritating and confusing for beginners .
- **Structure of Programs**
 - A C program is a **collection of functions**.
 - Execution starts at the **main()** function.



General structure of a C Program

Documentation Section

Link Section

Definition Section

Global Declaration Section

Function Section

Main()

{

____DECLARATION PART____

____EXECUTABLE PART____

}

Subprogram Section

Function 1

Function 2

..

..

```
/*This is my First program*/
```

```
#include<stdio.h> /*link */
```

```
void main()
```

```
{
```

```
printf("This my first C prog\n");
```

```
}
```



General structure of a C Program

Documentation Section

Link Section

Definition Section

Global Declaration Section

Function Section

Main()

{

____DECLARATION PART____

____EXECUTABLE PART____

}

Subprogram Section

Function 1

Function 2

..

..

```
/*this is my second program*/
```

```
#include<stdio.h> /*link */
```

```
#define pi 3.14 /*definition*/
```

```
float area; /*global */
```

```
void main()
```

```
{
```

```
float r; /*Declaration*/
```

```
r=1.0;
```

```
area=pi*r*r;
```

```
printf("area is %d \n",area);
```

```
}
```



Writing and executing a C Program in LINUX

Create a c program

cd to your directory of choice, and write the c-program using some text editors (eg. vi,vim)

e.g. **cd** ~/Cprog
 vim firstprog.c

Compiling and linking

use an existing compiler (gcc, cc etc) to compile your program and make an executable

e.g. **cc** *filename*
 gcc *filename1, filename2, filename3*

Common options:

-lm : library math

-c : Compiles without linking. binaries are saved as "*filename.o*"

-o *exename*: compiled binary with specified filename. (*a.out* default)

gcc -o myprog firstprog.c

Execute the program

./filename will execute the program. (Should have the “**x**” flag on !)



C variables and Data Types




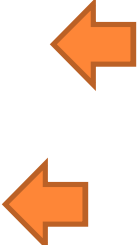
Keywords:

Keywords Serve as Basic building blocks of the programs.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Figure 1.5


```
#include <stdio.h>
#define PI 3.14
float area;
void main()
{
    double r;
    r=1.0;
    area=PI*r*r;
    printf("Area is %d \n",area);
}
```



Backslash character constants:

Character	Meaning
'\a'	alert (bell)
'\b'	backspace
'\f'	form feed
'\n'	newline
'\r'	carriage return
'\t'	horizontal tab
'\v'	vertical tab
'\\'	backslash
'\''	single quote
'\"'	double quote
'\?'	question mark

```
#include <stdio.h>
#define PI 3.14
float area;
void main()
{
    double r;
    r=1.0;
    area=PI*r*r;
    printf("Area is %d \n", area);
}
```



Variables

a variable declaration reserves and names an area in memory at run time to hold a value of particular type.

- C puts the type first followed by the name.
- They begin with a letter
- Upto 31 characters (ANSI)
- Case Sensitive
- **Must not** be a keyword
 - **Should not** conflict with built-in functions
- No while spaces
- Retains random values unless set.

```
/*This is my second program*/

#include "stdio.h" /*link */

#define PI 3.14 /*definition*/

float area;

void main()
{
  double r;
  r=1.0;
  area=PI*r*r;
  printf("Area is %d \n",area);
}
```



Data Types

Integers

All behave like integers and can be mixed with one another.: e.g. `int`, `short`, `long`, `char`

Floating Point:

`float`(~6 digit precision), `double`(~15 digit precision), `long double`

Variable Type	Keyword	Bytes Required	Range
Character	<code>char</code>	1	-128 to 127
Unsigned character	<code>unsigned char</code>	1	0 to 255
Integer	<code>int</code>	2	-32768 to 32767
Short Integer	<code>short int</code>	2	-32768 to 32767
Long Integer	<code>long int</code>	4	-2,147,483,648 to 2,147,438,647
Unsigned Integer	<code>unsigned int</code>	2	0 to 65535
Unsigned Short Integer	<code>unsigned short int</code>	2	0 to 65535
Unsigned Long Integer	<code>unsigned long int</code>	4	0 to 4,294,967,295
Float	<code>float</code>	4	1.2E-38 to
Double	<code>double</code>	8	2.2E-308 to
Long Double	<code>long double</code>	10	3.4E-4932 to 1.1E+4932

8 bit = 1 byte (Note: Actual byte/ datatype may vary from machine to machine)

Declaration syntax

data-type v1, v2,vn;

Variables are separated by commas.

Statement ends with a semicolon(;).

Example:

int count;

int a=5, b=2, c;

double ratio;

```
/*This is my second program*/
```

```
#include "stdio.h" /*link */
```

```
#define PI 3.14 /*definition*/
```

```
float area;
```

```
void main()
```

```
{
```

```
double r;
```

```
r=1.0;
```

```
area=PI*r*r;
```

```
printf("Area is %d \b", area);
```

```
}
```

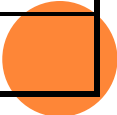


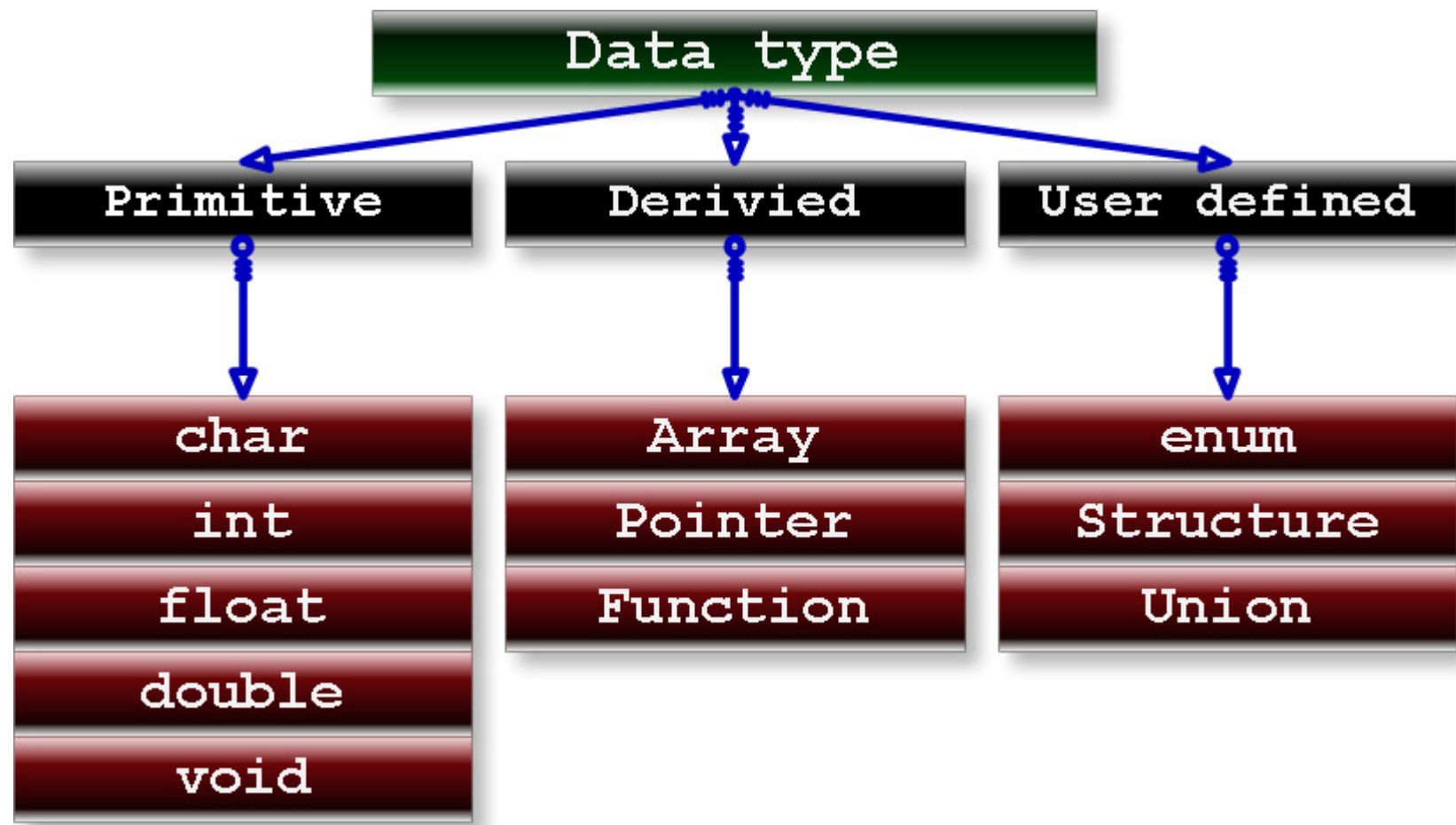
Format Specifies



```
printf("Area is %d \n", area);
```

<code>%i , %d %u %6d</code>	Integer(decimal, hexa or octa decimal) Decimal integer. Unsigned decimal integer. decimal integer, at least 6 characters wide.
<code>%f %6f %.2f %6.2f</code>	print as floating point floating point, at least 6 characters wide floating point, 2 characters after decimal point floating point, at least 6 wide and 2 after decimal point
<code>%o</code>	Octal
<code>%x</code>	Hexadecimal
<code>%c or %s</code>	Character / String
<code>%%</code>	For printing %
<code>%e,%E</code>	Exponential notation





TYPE COMBINATION AND PROMOTION

- The integral types may be mixed together in arithmetic expressions since they are all basically just integers with variation in their width.
- For example, **char** and **int** can be combined in arithmetic expressions such as ('b' + 5).
- In such a case, the compiler "promotes" the smaller type (char) to be the same size as the larger type (int) before combining the values.
- Promotions are determined at compile time based purely on the types of the values in the expressions.
- Promotions do not lose information -- they **always convert from a type to compatible, larger type** to avoid losing information.
- Overflow:



TRUNCATION

- The opposite of promotion, truncation moves a value from a type to a smaller type.
- The compiler just drops the extra bits. (May or may not generate a compile time warning of the loss of information).

Assigning from an integer to a smaller integer (e.g.. long to int, or int to char) drops the most significant bits.

```
char ch;  
int i;  
i = 321;  
ch = i;  
//int → char (ch is now 65)
```

Assigning from a floating point type to an integer drops the fractional part of the number.

```
double pi;  
int i;  
pi = 3.14159;  
i = pi;  
// float→int (i is now 3)
```



MORE EXAMPLES

scale the unit test marks in the range 0 - 20 to the range 0 -100.

Divide the marks by 20, and then multiply by 100 to get the scaled marks.

```
#include<stdio.h>
main()
{
    int score;

    scanf("enter the marks:  %d" &score);

    score = (score / 20) * 100;

    printf("revised marks= %d", score);
}
```

```
score = ((double)score / 20) * 100;
```



Operators and Expressions



THE ASSIGNMENT OPERATORS

=	<p>Assignment operator (copies the value from its right hand side to the variable on its left hand side)</p> <p>The assignment also acts as an expression which returns the newly assigned value</p>	<pre>i = i + 3;</pre> <p>(means i ← i + 3)</p> <pre>y = (x = 2 * x);</pre>
Other Assignment operators besides the “ = ” operator are as follows:		
+= , -=	Increment or decrement by RHS	i += 3
*= , /=	Multiplication or division by RHS	<pre>i /= 3</pre> <p>is equivalent to <pre>i = i / 3</pre></p>
>>= , <<=	Bitwise Right/Left Shift by RHS (divide/Multiply by 2 ⁿ)	
&= , = , ^=	Bitwise AND,OR,XOR by RHS.	

MATHEMATICAL OPERATORS

C supports the following operation

+	Addition
-	Subtraction
/	Division
*	Multiplication
%	Remainder (mod)

- use parenthesis to avoid any bugs due to a misunderstanding of precedence.
- For example, A division (/) with two integer arguments will do integer division. If either argument is a float, it does floating point division. So (10/4) evaluates to 2 while (10/4.0) evaluates to 2.5.



SIMPLE CONVERSIONS

$$(1) y = mx^2 \quad (2) y = 10^{-4} e^{-\sin(x)^2/5} \quad (3) y = \frac{a+bx}{x \sin(x^2)}$$

$$(4) y = \sqrt{\sigma \left(\frac{1+x^2}{|1-x|} \right) \sqrt{\log_{10}(1+p^2)} + 6} - x$$

$$(1) y = m * x * x;$$

$$(2) y = 1.0E-4 * \exp(-\text{pow}(\sin(x), 2) / 5.0);$$

$$(3) y = (a + b * x) / (x * \sin(x * x));$$

$$(4) y = \text{sqrt}(\text{sigma}(1 + x * x) / \text{fabs}(1 - x) * \text{sqrt}(\log_{10}(1 + p * p)) + 6) - x;$$

or

$$a = \text{sqrt}(\log_{10}(1 + p * p));$$

$$b = \text{sigma}(1 + x * x) / \text{fabs}(1 - x);$$

$$y = \text{sqrt}(a * b + 6) - x;$$



UNARY OPERATORS

Increment (++) and decrement(--) operators

The increment (decrement) operator increases (decreases) the value of the variable by 1.

```
int i=42, j;
```

Operation	Example	Equivalent to	
var++ (post increment)	j=(i++ + 10);	j = i+10; i = i+1;	j is 52
++var (pre increment)	j= (++i +10);	i = i+1; j= i+10;	j is 53
Var-- (post decrement)	j=(i-- + 10);	j = i+10; i = i-1;	j is 52
--var (pre decrement)	j=(--i + 10);	i = i-1; j= i+10;	j is 51

RELATIONAL OPERATORS

These operate on integer or floating point values and return a 0 or 1 boolean value.

Operator	Meaning
<code>==</code>	Equal
<code>!=</code>	Not equal
<code>></code>	Greater Than
<code><</code>	Less Than
<code>>=</code>	Greater or Equal
<code><=</code>	Less or Equal
<code>=</code>	ASSIGNMENT operator (does not belong to this class)



LOGICAL OPERATORS

Logical operators take **0 as false** and **anything else to be true**

BOOLEAN operators	
Op	Meaning
!	Boolean NOT
&&	Boolean AND
	Boolean OR

Bitwise operators have higher precedence over Boolean operators.

BITWISE OPERATORS (Manipulates data at BIT level)	
Op	Meaning
~	Bitwise Negation
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR
>>	Right shift (divide by power of 2)
<<	Left shift (divide by power of 2)

STANDARD LIBRARY FUNCTIONS

*basic housekeeping functions are available to a C program in form of standard library functions. To call these, a program must **#include the appropriate .h file***

stdio.h	file input and output
math.h	mathematical functions such as sin() and cos()
conio.h/ cursers.h	
time.h	date and time
ctype.h	character tests
string.h	string operations
stdlib.h	utility functions such as malloc() and rand()
assert.h	the assert() debugging macro
stdarg.h	support for functions with variable numbers of arguments
setjmp.h	support for non-local flow control jumps
signal.h	support for exceptional condition signals
limits.h, float.h	constants which define type range values such as INT_MAX

<http://www.cplusplus.com/reference/cmath/>

(only important contents shown, for all contents search the web)

math.h contents

Trigonamitric/ hyperbolic: `cos, sin, tan, acos, asin, atan, cosh, sinh, tanh`

Exponential and logarithmic functions: `exp, log, log10`

Power functions: `pow, sqrt`

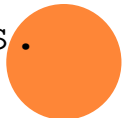
Rounding and remainder etc: `ceil, floor, fmod, fabs`

stdio.h contents

Formatted input/output: `fprintf, fscanf, printf, scanf, sprintf, sscanf.`

File access: `fclose, fopen, fread, fwrite.`

Character i/o: `fgetc, getc, getchar, gets, putc, putchar, puts.`



<http://www.cplusplus.com/reference/cmath/>

(only important contents shown, for all contents search the web)

Stdlib.h contents

Pseudo-random sequence generation: `rand`, `srand`

Dynamic memory management (DMA): `calloc`, `free`, `malloc`, `realloc`

Environment: `abort`, `exit`, `getenv`, **`system`**

string.h contents

`strcpy`, `strcat`, `strcmp`, `strchr`, `strlen`

time.h contents

`Clock`, `difftime`, `time`

ctype.h contents

`isalnum`, `isalpha`, `isblank`, `iscntrl`, `isdigit`, `islower`, `isupper`



THE C PREPROCESSOR

- The Preprocessor is a program that processes the C code before it is processed by the compiler.
- Preprocessor Directives are placed before the main()
- Common Directives: `#define`, `#include`, `#undef`, `#ifdef`, `#endif`, `#ifndef`, `#if`, `#else`

#define <i>Sets up symbolic textual replacements in the source.</i>	#include <i>Brings in texts from different files during compilation.</i>
<pre>#define PI 3.14 #define Test if(x>y) #define cube(x) x*x*x</pre>	<pre>#include<file1.h> // System #include "file1.h" //user</pre>



ASSIGNMENT -I

TO BE SUBMITTED BY 19-08-2013.

THIS IS A PART OF CONTINUOUS ASSESSMENT, AND WILL CARRY MARKS

SUBMIT THEM IN A SHEET BY 19-08-2013. WRITE:

YOUR NAME, LEVEL (MSc / NANO/ IMSc ETC) AND ROLL NO

1. Convert to equivalent C statement

(a) x^{y^z}

(b) $\frac{ax + b/c}{ax - bc}$

(c) $\frac{a}{bc + |d|} - f$

(d) $\sqrt{\sigma\left(\frac{1+x}{1-x}\right)}\sqrt{1-y} + 6 - x$

(e) $\frac{1}{\alpha\sqrt{2\pi kT}} e^{\sqrt{2\sigma(x-m)kT}}$



ASSIGNMENT-I (CONTD..)

- (2) Write Programs to,
- a. Display your name.
 - b. add two given nos ($c=a+b$) and display the results
 - c. Read two integers and display the result of their division($c=a/b$).
 - d. Calculate radius of a circle, given its area. Display your result only upto 3 decimal places.
 - e. write a program to display various numbers associated with the ASCII characters.

PRACTICE THEM DURING LAB ON TUESDAY (AND OTHER DAYS AS WELL) TO ENSURE THE CORRECTNESS OF THE PROGRAM SUBMITTED.

