A Quick Reference to C Programming Language

Structure of a C Program

```
#include(stdio.h)                    /* include IO library  */
#include...                         /* include other files */
#define..                           /* define constants    */

/* Declare global variables*/)
(variable type)(variable list);

/*  Define program functions */
(type returned)(function name)(parameter list)
(declaration of parameter types)
 {
   (declaration of local variables);
   (body of function code);
 }
/* Define main function*/
main ((optional argc and argv arguments))
(optional declaration parameters)
   {
   (declaration of local variables);
   (body of main function code);
   }
```

Comments

```
Format:     /*(body of comment)     */
Example:    /*This is a comment in C*/
```

Constant Declarations

```
Format:     #define(constant name)(constant value)
Example:    #define MAXIMUM 1000
```

Type Definitions

```
Format:      typedef(datatype)(symbolic name);
Example:     typedef int KILOGRAMS;
```

Variables

Declarations:

```
Format:     (variable type)(name 1)(name 2),...;
Example:    int firstnum, secondnum;
```

```
char alpha;
int firstarray[10];
int doublearray[2][5];
char firststring[1O];
```

Initializing:

```
Format:     (variable type)(name)=(value);
Example:    int firstnum=5;
```

Assignments:

```
Format:     (name)=(value);
Example:    firstnum=5;
            Alpha='a';
```

Unions

Declarations:

Format:

```
union(tag)
{(type)(member name);
 (type)(member name);
     ...
}(variable name);
```

Example:

```
union demotagname
{int a;
 float b;
}demovarname;
```

Assignment:

Format:

```
(tag).(member name)=(value);
demovarname.a=1;
demovarname.b=4.6;
```

Structures

Declarations:

Format:

```
struct(tag)
{(type)(variable);
 (type)(variable);
     ...
}(variable list);
```

Example:

```
struct student
{int idnum;
 int finalgrade;
 char lettergrade;
} first,second,third;
```

Assignment:

Format:	`(variable name).(member)=(value);`
Example:	`first.idnum=333;`
	`second.finalgrade=92;`

Operators

Symbol	Operation	Example
`+,-,*,/`	arithmetic	`1 = b + c;`
`%`	mod	`a = b % c;`
`>`	greater than	`if (a > b)`
`>=`	greater than or equal	`if (a >= b)`
`<`	less than	`if (a <b)`
`<=`	less than or equal	`if (a <= b)`
`==`	equality	`if ( == b)`
`=`	assignment	`a=25;`
`!=`	not equal	`if (a != b)`
`!`	not	`if (!a)`
`&&`	logical and	`if (a) && (b)`
——	logical or	if (a) —— (b)
`++`	increment	`++ a;`
`--`	decrement	`-- a;`
`&`	bitwise and	`a = b & c;`
—	bitwise or	a = b — c;
$\wedge$	bitwise xor	$a = b \wedge c$
`>>`	shift-right	`a = b >> 2;`
`<<`	shift-left	`a = b << 2;`
`~`	one's complement	`a = ~b`

Input and Output

Output

Print Formats:

`String:`	`print("(literal string)");`
`String+newline:`	`print ("(string)\n");`
`Variables:`	`printf("(conversion specs)",(variables));`

Print Examples:

```
print("firstvar+secondvar=%d\n",thirdvar);
```

Print Conversion Specifications:

`%d`	decimal
`%u`	unsigned decimal
`%o`	octal

`%h`	`hex`
`%e`	`exponential`
`%f`	`float`
`%g`	`shorter of %e or %f`
`%c`	`char`
`%s`	`string`

Print Escape Sequences:

`\n`	`newline`
`\t`	`tab`
`\r`	`carriage return`
`\f`	`form feed`
`\b`	`backspace`
`\'`	`output`
`\\`	`output \`

Input:

Scanf Format:

```
scanf("(conversion specs)",&(varl),&(var2),…);
```

Scanf Example:

```
scanf("%d %d %d",&first,&second,&third);
```

Scanf Conversion Specifications:

`%d`	`decimal integer expected`
`%o`	`octalinteger expected`
`%x`	`hex integer expected`
`%h`	`short integer expected`
`%c`	`character expected`
`%s`	`string expected`
`%r`	`real value expected`
`%e`	`exponential notation expected`

Primitive Input and Output Examples:

Get a character from standard input:	`c = getchar();`
Put a character on standard output:	`putcher(c);`

Control Structures

FOR LOOP Format:

```
for  ((first expr);(second expr);(third expr))
     (simple statement);
for  ((first expr);(second expr);(third expr))
     {
     (compound statement);
```

```
        }
```

WHILE LOOP Format:

```
while   ((condition))
        (simple statement);
while   ((condition))
        {
          (compound statement);
        }
```

DO WHILE LOOP Format:

```
do
        (simple statement)'
while   ((condition))
do      {
         (compound statement);
        }
while   ((condition));
```

IF CONDITIONAL Format:

```
if      ((condition))
        (simple statement);
if      ((condition))
        {
         (compound statement);
        }
```

IF... ELSE CONDITIONAL Format:

```
if      ((condition))
        (statement 1);
else
        (statement 2);
```

SWITCH Format:

```
switch  ((expression))
        {case (value 1):(statement 1);
         case (value 2):(statement 2);
         ...
         default:(default statement);
        }
```

Function Definitions

Format:

```
(type returned)(function name)((parameter list))
(declaration of parameter list variables)
{
  (declaration of local variables);
  (body of function code);
}
```

Example:

```
   Int. adder(a,b)
   int a,b;
     {int c;
      c = a + b;
      return (c);

     }
```

Pointers

Declaration of pointer variable:

```
    Format:       (type)*(variable name);
    Examples:     int *p;
                  struct student *classmember;
```

The major ingradients of C Programming language:

A *C* program consists of a *main function* and several *program functions*. The program can also access many *external functions* that are contained in the *header file* and *C library*.

- The roles of the *main function* include declaring global variables, defining program functions and specifying the sources of external functions.
- The *header file* normally contains frequently used utility functions such as IO library, etc.
- The *program function* carries out a specific task of the program, acting as a building block of the program. Arguments can be used to pass values. The name of the function can also be used as a variable of specified type to return a value to the main program.
- An array is indexed by a pointer. The pointer starts at 0, rather than 1.

In the simple tutorial of *Introduction to C Programming*, we will learn the very basic elements of a C program through an example. To under each elements of this short program and try to add additional features to the program.