A Quick Reference to C Programming Language

Structure of a C Program

```
#include Name of library                  /* include IO library   */
#include <stdio.h>                        /* include other files */

#define max 10                            /* define constants    */

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

/* Define program functions */
(type returned)(function name)(parameter list)
{
   (declaration of local variables);
   (body of function code);
 }

/* Define main function*/
Void main ()
   {
   (declaration of local variables);
   (body of main function code);
   }
```

Comments

```
    Format:
         1. /*(body of comment) */
         2. //(body of comment)

    Example:    /*This is a comment in C*/
```

Constant Declarations

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

Variables

Declarations:

```
Format:      (variable type)(name 1),(name 2),…;

Example:     int firstnum, secondnum;
             char alpha;

             int firstarray[10];
             int oublearray[2][5];
             char firststring[10];
```

Initializing:

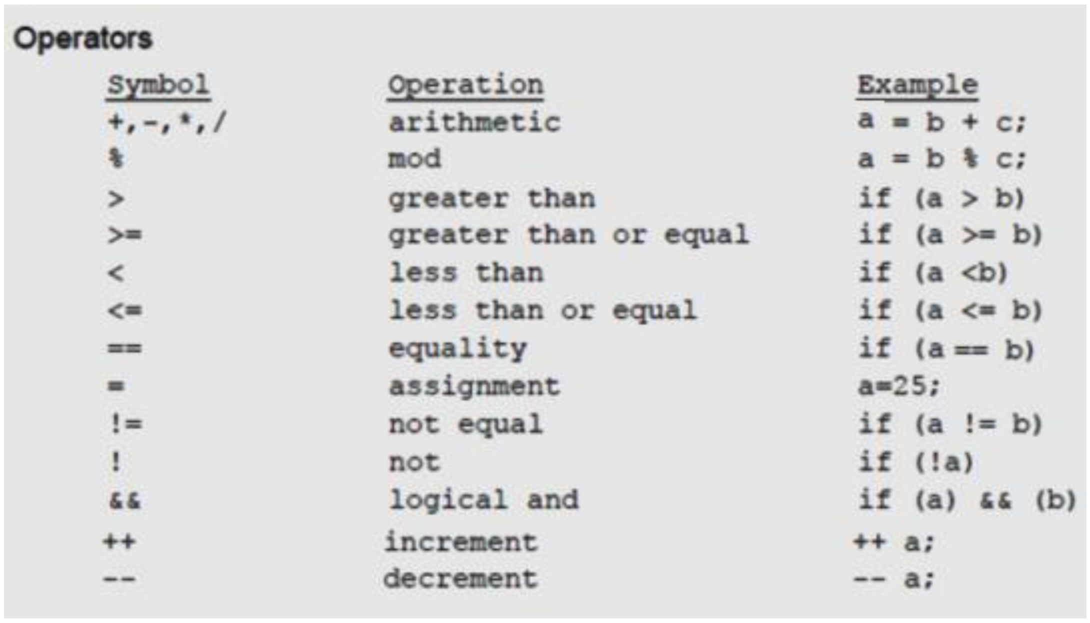
```
Format:      (variable type)(name)=(value);
Example:     int firstnum=5;
```

Assignments:

```
Format:      (name)=(value);

Example:

    firstnum=5;
    Alpha='a';
```



Operators

Symbol	Operation	Example
`+,-,*,/`	arithmetic	`a = 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 (a == b)`
`=`	assignment	`a=25;`
`!=`	not equal	`if (a != b)`
`!`	not	`if (!a)`
`&&`	logical and	`if (a) && (b)`
`++`	increment	`++ a;`
`--`	decrement	`-- a;`

Input and Output

Output

Print Formats:

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

Print Examples:

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

Print Conversion Specifications:

%d	decimal
%f	float
%c	char
%s	string

Print Escape Sequences:

\n	newline
\t	tab
\b	backspace
\'	output '
\\	output \

Input:

Scanf Format:

```
scanf("(conversion specs)",&(var1),&(var2),…);
```

Scanf Example:

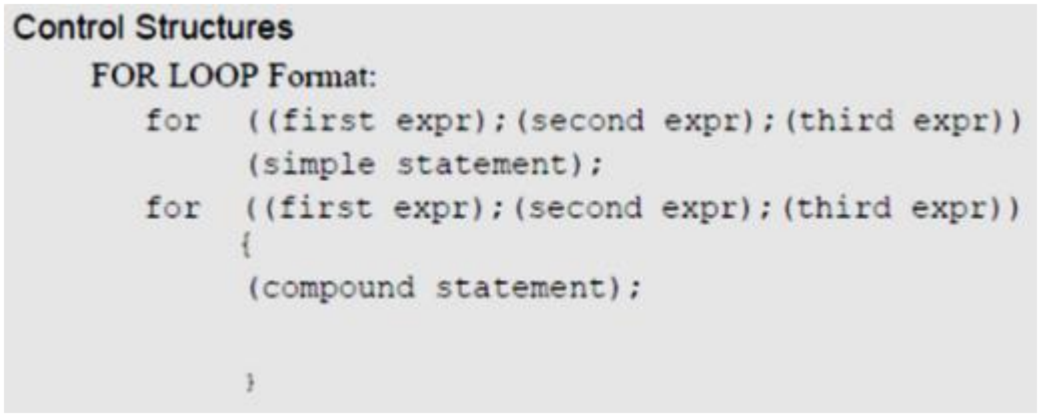
```
scanf("%d %d %d",&first,&second,&third);
```

Scanf Conversion Specifications:

```
%d   decimal integer expected
%c   character expected
%s   string expected
%r   real value expected
```

Primitive Input and Output Examples:

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

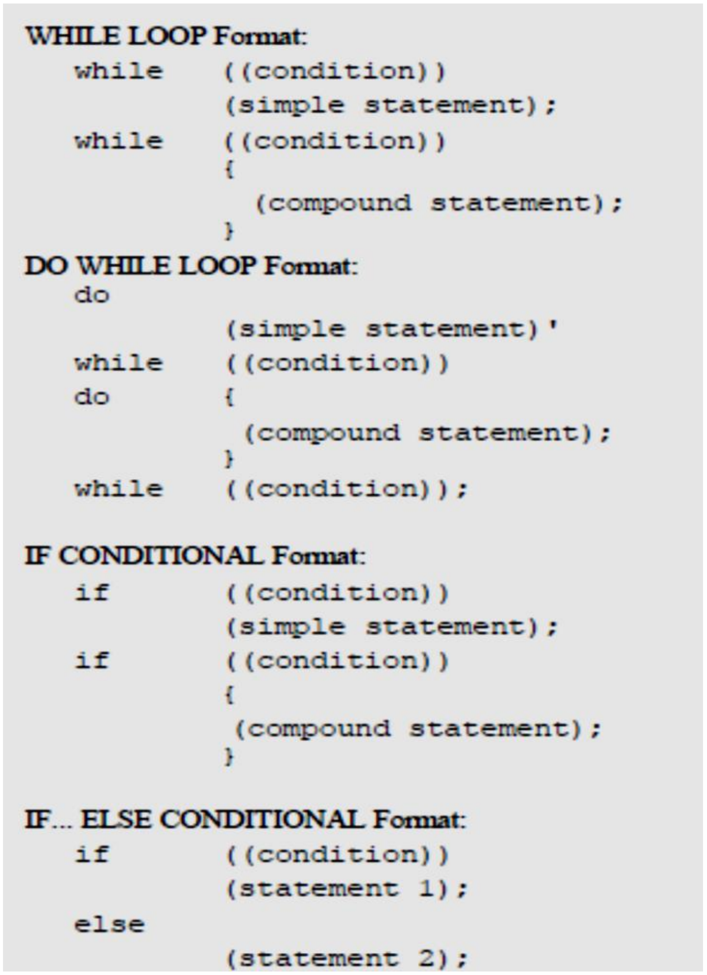


Control Structures

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);
           }
```

Structures

Declarations:

Format:

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

Note you need to define an parameter that its type is struct(tag)

OR

Format:

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

Example

```
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book;
```

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;
```

Example:

```
#include <stdio.h>
#include <string.h>

struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} Book1, Book2;

int main( ) {



   /* book 1 specification */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* print Book1 info */
   printf( "Book 1 title : %s\n", Book1.title);
   printf( "Book 1 author : %s\n", Book1.author);
   printf( "Book 1 subject : %s\n", Book1.subject);
   printf( "Book 1 book_id : %d\n", Book1.book_id);

   /* print Book2 info */
   printf( "Book 2 title : %s\n", Book2.title);
   printf( "Book 2 author : %s\n", Book2.author);
   printf( "Book 2 subject : %s\n", Book2.subject);
   printf( "Book 2 book_id : %d\n", Book2.book_id);

   return 0;
}
```

References

[1] Dr. Armando Tura, A Quick Reference to C Programming Language.