C Language

If

```c
#include <stdio.h>

int main () {
   int a = 100;

   if( a == 10 ) {
      printf("Value of a is 10\n" );
   }
   else if( a == 20 ) {
      printf("Value of a is 20\n" );
   }
   else if( a == 30 ) {
      printf("Value of a is 30\n" );
   }
   else {
      printf("None of the values is matching\n" );
   }

   printf("Exact value of a is: %d\n", a );

   return 0;
}
```

While

```c
#include <stdio.h>

int main () {
```

```
   int a = 10;
   while( a < 20 ) {
      printf("value of a: %d\n", a);
      a++;
   }

   return 0;
}
```

For

```
#include <stdio.h>

int main () {

   int a;
   for( a = 10; a < 20; a = a + 1 ){
      printf("value of a: %d\n", a);
   }

   return 0;
}
```

Switch

```
#include <stdio.h>

int main ()

   char grade = 'B';

   switch(grade) {
      case 'A' :
```

```
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
         break;
      default :
         printf("Invalid grade\n" );
   }
   return 0;
}
```

Struct

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

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

int main( ) {

   struct Books Book1;        /* Declare Book1 of type Book */
   struct Books Book2;        /* Declare Book2 of type Book */

   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;
```

```
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   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;
}
```

Functions

```
int max(int num1, int num2) {

   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

Pointers

```c
#include <stdio.h>

int main () {

   int  var = 20;   /* actual variable declaration */
   int  *ip;        /* pointer variable declaration */

   ip = &var;  /* store address of var in pointer variable*/

   printf("Address of var variable: %x\n", &var  );

   /* address stored in pointer variable */
   printf("Address stored in ip variable: %x\n", ip );

   /* access the value using the pointer */
   printf("Value of *ip variable: %d\n", *ip );

   return 0;
}
```

Aritmética de Ponteiros

```c
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have array address in pointer */
   ptr = var;

   for ( i = 0; i < MAX; i++) {
```

```
        printf("Address of var[%d] = %x\n", i, ptr );
        printf("Value of var[%d] = %d\n", i, *ptr );

        /* move to the next location */
        ptr++;
    }

    return 0;
}
```

Depois de executar:

```
Address of var[0] = bf882b30
Value of var[0] = 10
Address of var[1] = bf882b34
Value of var[1] = 100
Address of var[2] = bf882b38
Value of var[2] = 200
```

Data Types

```
            char  c0 = 'A';  // by default signed on most compilers
  signed    char  c1 = 'B';  // make sure the type is signed
unsigned    char  c2 = 'C';

            short s0 = 1763;  // the same as signed short
unsigned    short s1 = 1728;

            int   i0 = -1373762;  // the same as signed int
unsigned    int   i1 = 8382382U;  // the trailing U signals that the integer constant is unsigned

            long  l0  = 82781762873L;  // the same as signed long and signed long int
unsigned    long  l1 = 38273827322UL;  //   the int is optional, so we do usually do not put it

        long long  L0  = 8278176284398439847 3LL;  // the same as signed long long int
unsigned long long  L1 = 3827382733493498332 2ULL;  //   the int is optional
```

Typedef

In C it is also possible to give another name to an existing data type using the typedef keyword. For example, the following code fragment declares a data type named u64 that is supposed to be a 64-bit

unsigned integer:

```
#ifdef IS_A_32_BIT_CPU
typedef unsigned long long u64; // A 64-bit data type on a 32-bit CPU
#endif
#ifdef IS_A_64_BIT_CPU
typedef unsigned long u64;      // A 64-bit data type on a 64-bit CPU
#endif
```

The rest of our code can now use the type u64. Switching from a 32-bit to a 64-bit CPU requires only two very small changes in the code (and a recompilation), namely, undefining the symbol IS_A_32_BIT_CPU and defining the symbol IS_A_64_BIT_CPU.

Arrays

Using an out-of-range value does not result in any compiler error but will usually lead to a hard to discover run-time error.

```
int a[100];
int *pa = &a[30]; // same as int *pa = a + 30;
int *pA = &a[-2]; // same as int *pA = a - 2;
```