Bit wise Operators

Bit wise Operators

- These operators are used to perform bit operations. Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits
- Bit wise operators in C language are
  - & (bitwise AND)
  - | (bitwise OR)
  - ~ (bitwise NOT),
  - << (left shift)
  - >> (right shift)

TRUTH TABLE FOR BIT WISE OPERATION & BIT WISE OPERATORS

x	y	x|y	x&y	x^y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

- Consider x=40 and y=80. Binary form are :
  - x = 00101000
  - y= 01010000
- All bit wise operations for x and y are given below:
- x&y = 00000000 (binary) = 0 (decimal)
- x|y = 01111000 (binary) = 120 (decimal)
- ~x = 11010111 = -41 (decimal)
- x << 1 = 01010000 (binary) = 80 (decimal)
- x >> 1 = 00010100 (binary) = 20 (decimal)

C Preprocessor

The C Preprocessor

- The C preprocessor is a program that processes any source program in C before compilation.
- It allows the user to define macros, the C preprocessor is also called *a macro processor*.
- A *macro* is defined as an open-ended subroutine. An open-ended subroutine is a set of program instruction, as in a function, that does not have a return statement.
- The preprocessor provides its own language that can be a very powerful tool for the programmer. These tools are instructions to the preprocessor, and are called *directives*.
- The C preprocessor has several directives that are used to invoke it.
- A directive usually occupies a single line. The # symbol should be first non-blank character on the line.

The C preprocessor Directives

- The preprocessor directives can be classified into two categories:
  Unconditional and conditional
- Unconditional:
  define, undef, include, line, error, pragma
- Conditional
  if, else, elif, ifdef, ifndef, endif

#define: Defines a macro
#undef: Undefines a macro
#include: Textually includes the contents of a file.
#ifdef: Makes compilation of code conditional on a macro being defined
#ifndef: Makes compilation of code conditional on a macro not being defined
#endif: Marks the end of a conditional compilation block.
#if: Makes compilation of code conditional on an expression being non-zero.
#else: Specifies an else part for a #ifdef, #ifndef, or #if directive
#elif: Combination of #else and #if
#line: change current line number and file name
#error: Outputs an error message
#pragma: is implementation-specific

The C preprocessor Directives

- #define general form:
  #define macro_name replacement_string
- #define directive is used to make substitution throughout the program in which it is located. #define causes the compiler to go through the program, replacing every occurrence of macro_name with replacement_string.

e.g

```
#include<stdio.h>
#define abs_value(a) ((a<0)? –a : a)
main()
{
    int a=-1;
    while (abs_value(a))
    {
        printf("\n Value of a = %d within while", a);
        a=0;
    }
    printf("\n Value of a=%d outside while", a);
}
```

Output: Value of a=-1 within while
Value of a=0 outside while

Macros

- Macro substitution works within macros also !!!
- E.g:

```
#define num_records 100
#define record_size 1024
#define relation_size record_size*num_records
```

// now *relation_size* has value 102400 and this will be substituted in the program.

Macros (contd.)

- String replacement with argument passing
- *#define identifier(x,y) string*
- A working example:

```
# include <stdio.h>
# define SWAP(X,Y) {int temp; temp = X; X = Y; Y = temp;}
# define PRINT(P,Q) {printf ("a = %d, b = %d\n\n", P, Q);}
void main ()
{
  int a = 5, b = 14;
  SWAP(a,b);
  PRINT(a,b);
}
```

Macros (contd.)

- Two *intelligent* Macro operators:
  - Stringizing
    - The # operator converts actual args into string.
    - # define MESG(F) printf(#f)
    - MESG(testing mesg); → printf(“testing mesg”);
  - Concatenating
    - The ## operator concatenates 2 tokens
    - # define ERR(X,Y) printf(X ## Y);
    - ERR(“err: “, “this err”); → printf(“err: this err”);

Example

```
#include<stdio.h>
#define SWAP(X,Y) { int temp; temp=X;X=Y;Y=temp;}
#define PRINT(P,Q) {printf("a=%d, b=%d\n\n",P,Q);}
#define MESG(F) printf(#F)
#define ERR(X,Y) printf(X##Y)

void main()
{
    int a=5,b=14;
    SWAP(a,b);
    PRINT(a,b);
    MESG(testing mesg);
    ERR("err:","this err");
}
```

Conditional Compilation

- What are the directives ???
  - *#if, #elif, #else, #endif*
  - Behave exactly like their C counterparts.

Conditional Compilation

```
# include <stdio.h>
# define DEBUG 2
int main ()
{
   #if DEBUG == 2
     printf ("print this if DEBUG = 2\n");
   #else
     printf ("print this if DEBUG != 2\n");
   #endif
   return 0;
 }
```

- Helps to debug programs, as we can check parts of the program depending on the value of **DEBUG** (in our example).
- The identifier can be given any name.
- Every **#if** directive requires an **#endif** directive.