Preprocessor

What is preprocessor

- Program that pre-processes the source program before passing it on to the compiler
- Has set of commands which can be written in the code
- Provides a lot of convenience

Lets have a look…

Features

- Preprocessor commands are known as preprocessor directives
- Macro expansion
- File inclusion
- Conditional compilation
- Miscellaneous directives

Macro expansion

Eg: #define PI 3.1415

- Now in the entire program during pre processing every occurrence of PI will be replaced by 3.1415
- The above statement is called a ‘macro definition’ or more generally a ‘macro’
- PI is called macro template whereas 3.1315 is corresponding macro expansion

Advantages

- Makes code easier to read
- One change and it will be reflected across the program

Why not a variable?

- Compiler generates compact code for constants than for variables
- Using variable for what is a constant is sloppy
- It may get changed in the program

#define directive

- This can be used not only for constant but for complete statements also

Eg: #define ROCK printf(“We rock”);

This replaces every occurrence of ROCK with the printf statement

- We can pass arguments in Macros just like functions

Eg: #define AREA(x) (3.14*x*x);

a = AREA(5);

Remember

- No spaces between macro template and its argument

Eg: AREA(s) vs AREA (s)

- Enclose it in parentheses

Eg: #define SQUARE(n) n*n

j = 64/SQUARE(4);

Gives 64/4*4 = 64 against 4 that we were expecting

Remember cont..

- Macros can be split across multiple line with a \ at the end of each line

```
Eg: #define FORLOOP for(i=0;i<5;i++)\
                              printf("Hello");
```

- If you can't debug then look at macro expanded file for errors

Macros vs. functions

- Macros make program faster but increase size
- Functions make program smaller and compact
- Macro used 100 times will be expanded 100 times but function call 100 times will trigger the same function
- In short if small then macro but if big and complex functions must be used

File inclusion

- This directive causes one file to be included in other

Eg: #include “filename”

It causes content of filename to be inserted into source code at that point in the program

Uses:

- For large program to divide into files
- For some commonly used functions and macros

Conditional compilation

- Conditional compilation is done using #ifdef and #endif

Eg:

```
#ifdef macroname
        statement;
#endif
```

If macroname has been defined then statement will be executed; otherwise not

Uses of conditional compilation

- Used as a method to comment out code
- Used to make programs portable
- To avoid multiple inclusions of different files

#if, #else, #elif directives

- #iif is used to test value of an expression
- If the result of expression is non zero then the subsequent #else or #elif are compiled otherwise they are skipped

Eg: #if TEST<=5

```
        statement1;
     #else
        statement2;
      #endif
```

Miscellaneous directives

- #undef

Used to un-define a previously defined macro

- #pragma

Special purpose directive to turn on off certain features. They vary from compiler to compiler.

#pragma startup and #pragma exit

#pragma warn –rvl –par -rch

Summary

- The preprocessor directives enable the programmer to write programs that are easy to develop, read, modify and transport to a different computer system.
- We can make use of various preprocessor directives such as **#define**, **#include**, **#ifdef** - **#else** - **#endif**, **#if** and **#elif** in our program.
- The directives like **#undef** and **#pragma** are also useful although they are seldom used.

Questions

- Output:

```
#define SQR(x) (x*x)
main()
{
        int a,b=2;
        a=SQR(b+2);
        printf("%d\n",a);
        return 0;
}
```

- Output

```
#define FUN (i,j) i##j
main()
{
        int val1=10;
        int val12=20;
        printf("%d\n",FUN(val1,2));
        return 0;
}
```

- output

```
#define PRINT(int) printf(“int=%d”,int)
int main()
{
        int x=2,y=3,z=4;
        PRINT(x);
        PRINT(y);
        PRINT(z);
}
```

Answers

- 11
- 20
- int=2 int=3 int=4

Contact Info

- trainers@finaldesk.com
- rishabh@finaldesk.com
- nilesh@finaldesk.com
- jignesh@finaldesk.com
- yash@finaldesk.com
- anand@finaldesk.com