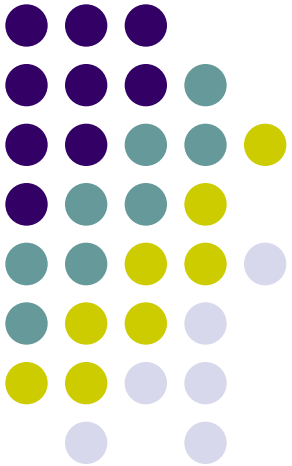
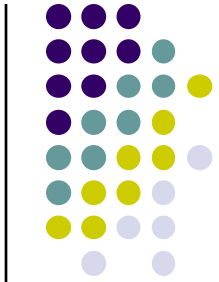
Paper 102: Programming & Problem solving through C

Lecture-09:Unit-I
C Fundamentals



The C Preprocessor

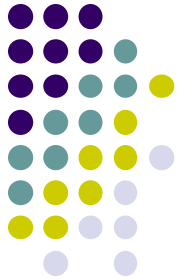


- A process called preprocessing is done on the source code before it is compiled.
- This is done by a program called the Preprocessor
- The preprocessor offers several features called preprocessor directives.
- Each of the directives start with a hash (#) sign .
- The line at the beginning of every C programs

  #include <stdio.h>

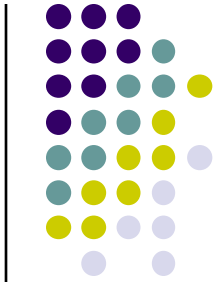
  Is a preprocessor directive

The C Preprocessor



- All lines preceding with a hash (#) sign are processed by the preprocessor
- The #include directive causes the preprocessor to effectively insert the *stdio.h* file into the C program, followed by the rest of the program
- Different types of preprocessor directives are:
  1. Macro expansion
  2. File inclusion
  3. Conditional compilation
  4. Miscellaneous

Macro expansion



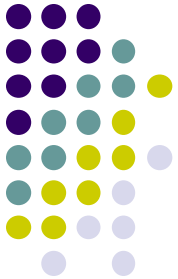
- The C preprocessor permits you to define simple macros that are evaluated and expanded prior to compilation

```
#define UPLIMIT 10
void main()
{       int I;
        for(i=1;i<=UPLIMIT;i++)
                printf("\n%d",i);
}
```

#define UPLIMIT 10

- This statement is called 'macro definition'/macro
- The preprocessor replaces every occurrence of UPLIMIT in the program by the value 10

Macro expansion example



```
#define AND &&
#define OR ||
void main()
{   int a=1,b=65,c=90;
    if((a<26) OR (b>=65 AND c<=90))
            printf(“This is a true statement”);
    else
            printf(“This is a false statement”);
}
```

It can also be used to replace the whole test expression

```
#define AND &&

#define ALPHABET (a>=65 AND a<=90)

void main()

{       int a=70;

        if(ALPHABET)

            printf("It is an alphabet");

}
```

Macro with arguments

```
#define SQUARE(n) (n*n)
void main()
{
      int a;
       a=64/SQUARE(4);
      printf("\n a=%d",a);
}
```

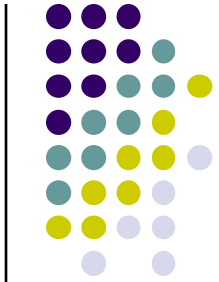
→ a=4

The whole macro expansion should be enclosed within parentheses

```
#define SQUARE(n) n*n
void main()
{
      int a;
       a=64/SQUARE(4);
      printf("\n a=%d",a);
}
```

→ a=64

Macro with arguments



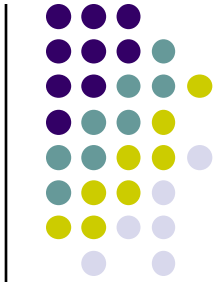
```
#define AREA(r) (3.14 * r * r)
void main()
{
    float r1=3.15,r2=2.5,a;
     a=AREA(r1);
    printf("\n Area of circle = %f",a);
    a=AREA(r2);
    printf("\n Area of circle = %f",a);
}
```

No space between Macro name and argument

After preprocessing: a=(3.14 * r1 * r1);

a=(3.14 * r2 * r2);

File Inclusion



1. This directive causes one file to be included in another.
2. The preprocessor command for file inclusion is written as

   #include “filename”

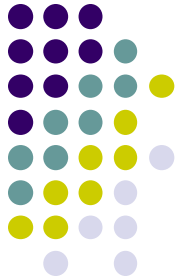
   #include<filename>

3. It can be use for a large program where declaration of variables and data structures are saved in a separate file
4. It can also be use for function definitions where they are saved in another file

This command would look for the file in the current directory and the same directories used for system header files

This command look for the file only in the specified list of directories

Conditional compilation

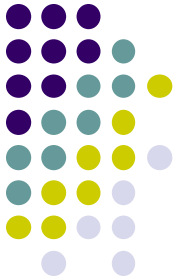


Using special preprocessing directives, you can include or exclude parts of the program according to various conditions.

Generally there are three kinds of reason to use a conditional macro.

- A program may need to use different code depending on the machine or operating system it is to run on.
- You may want to be able to compile the same source file into two different programs, where the difference between the programs is that one makes frequent time-consuming consistency checks on its intermediate data, or prints the values of those data for debugging, while the other does not.
- A conditional whose condition is always false is a good way to exclude code from the program but keep it as a sort of comment for future reference.

#ifdef -#endif and #ifdef-#else-#endif



```
#ifdef macroname
        statement 1;
        statement 2;
        statement 3;
#endif
```

- If macroname has been #define then the block of code will be processes as part of the program otherwise not.
- It is useful to comment out obsolete lines of code

```
#ifdef PCAT
    code suitable for a PC/AT
#else
    code suitable for a PC/XT
#endif
```

- To make programs portable and make them work in two totally different computers

#if -#else-#endif

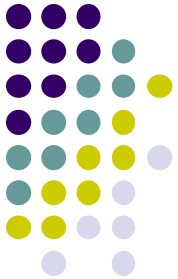
#if directive can be used to test whether an expression evaluates to a nonzero value or not.

```
void main()
{
   #if ADAPTER==MA
         codes for true block
   #else
         codes for false block
   #endif
}
```

#ifndef

This directive can be used to test whether a macro name has not been #define earlier

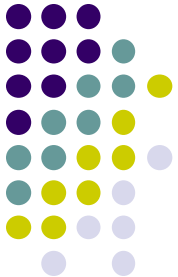
#if - #elif - #else - #endif



Nested testing can be made more compact with the help of the #elif directive

```
void main()
{
   #if ADAPTER==MA
        codes for monochrome adapter
   #elif ADAPTER==CGA
        codes for colour graphics adapter
  #elif ADAPTER==EGA
        codes for enhanced graphics adapter
  #elif ADAPTER==VGA
        codes for video graphics array
  #else
         codes for super video graphics array
  #endif
}
```

Miscellaneous Directives



Some of the preprocessor directives available

#undef **#error**

#pragma **#warning**

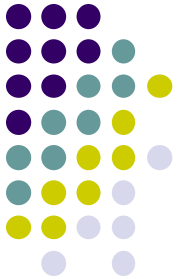
1. #undef is used to make a a macro which has been earlier #defined

   #undef PCAT

2. The #pragma is used to turn on or off certain features. Pragmas may vary from compiler to compiler.

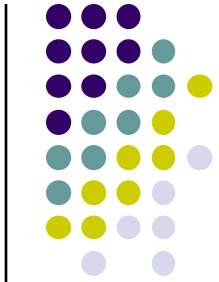
3. Turbo C compiler has got a pragma which allows assembly language statements to be included in C programs.

The #error and #warning Directives



1. The directive #error causes the preprocessor to report a fatal error. The rest of the line that follows #error is used as the error message.
2. The directive #warning is like the directive #error, but causes the preprocessor to issue a warning and continue preprocessing. The rest of the line that follows #warning is used as the warning message.

Class assignment



- Write a macro to check the greatest of three numbers. Use this macro in a program to determine the greatest of three numbers input by the user and display the result.
- Define macros for calculating the area of a circle, square and a triangle and store in a separate file called “areas.h”. Write a program to allow the user to choose which area to be computed and take appropriate inputs from the user to compute and display the result. Use “areas.h” in this program.