Vivekananda Institute
of Professional Studies

Preprocessor

Programming using C language
Dr. Nivedita Palia

Preprocessor Directives

A program that processes our program before it is passed to the compiler.

When the source code pass through ‘Preprocessor’ it is creates ‘ Expanded Source Code.

Start with # symbol.
Written at the top (in the beginning of program)
Written(usually) in CAPITAL letters(**convention**)

Types

1.Macro Expansion
2. File inclusion Directives
3.Compiler Controlled Directives / conditional compilation
4. Miscellaneous directives

Types of Macro

1. Simple
2. Argumented
3. Nested

Simple Macro Expansion

```
#include<stdio.h>
#define PI 3.14
void main()
{
float r=6.25,area;
area=PI*r*r;
printf("area of circle %f", area);
}
```

Macro Expansion

Macro template

Advantage: Not to change value at every place , only make change at one place and it will made changes in all occurrences.

Note: During Preprocessing , every macro template gets replaced with its corresponding macro expansion.

Argumented Macro Substitution

#define AREA(x) (3.14*x*x)

```
int main()

{

float r1 = 6.25, r2 = 2.5, a;

a = AREA (r1);

printf ("Area of circle = %f\n", a);

a = AREA (r2);

printf ("Area of circle = %f\n", a);

return 0;

}
```

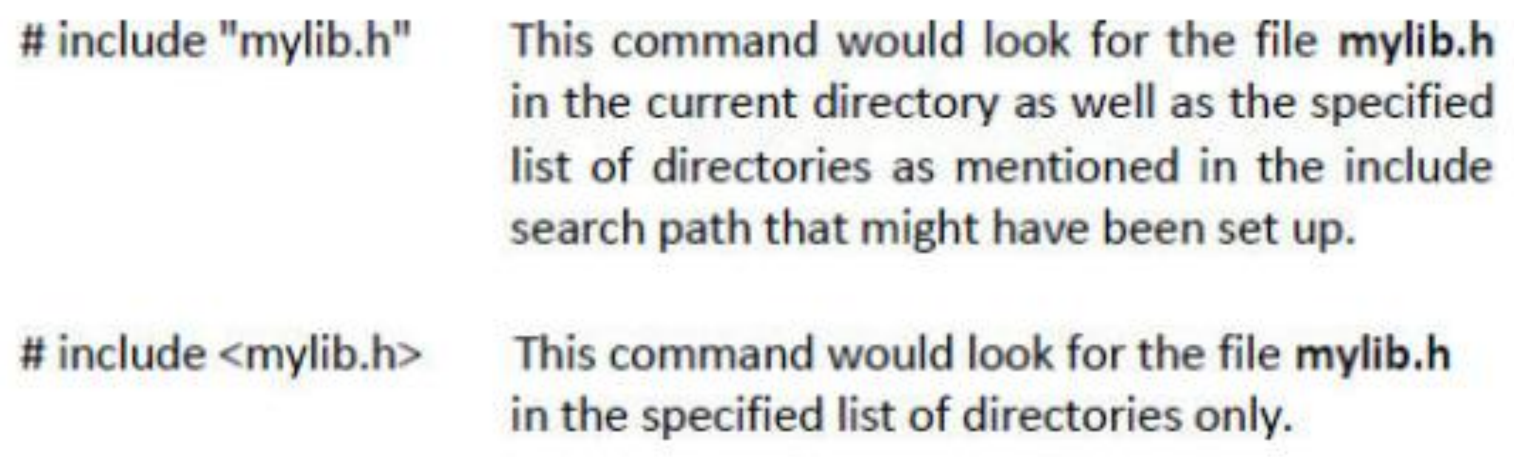
Nested Marco

```
#define SQUARE(x) (x*x)
#define CUBE(x) (SQUARE(x)*x)
```

Macro VS Function

Macro	Function
Macro are Preprocessed	Functions are compiled
Macro increased code length(space)	Size of the code remain same
Faster Execution	Slower Execution
During Preprocessing macro is replaced by macro value	During function call , control transfer from calling to called function

File inclusion directive



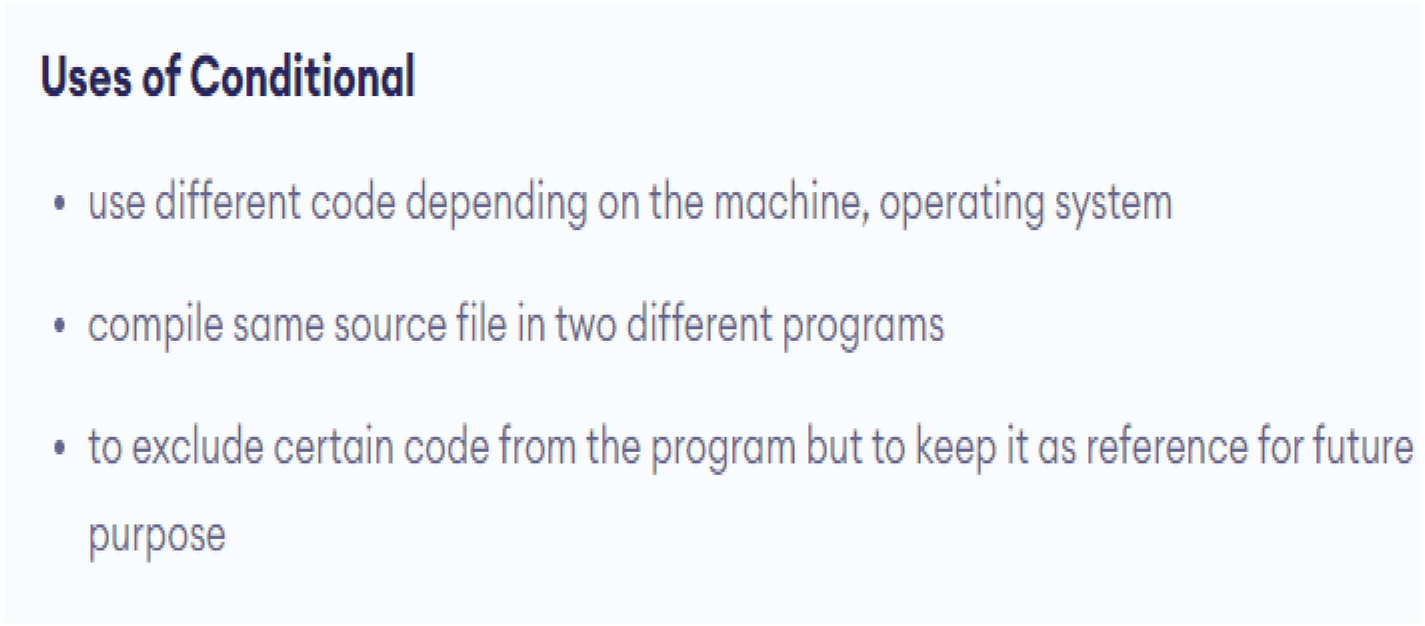
# include "mylib.h"	This command would look for the file **mylib.h** in the current directory as well as the specified list of directories as mentioned in the include search path that might have been set up.
# include <mylib.h>	This command would look for the file **mylib.h** in the specified list of directories only.

Compiler Controlled Directives / conditional compilation

- These directives are based on some conditions.
- Controlled by compiler.

- Types:
- #ifdef …… #endif
- #ifndef …… #endif
- #ifdef … #else … #endif
- #if …… #elif …… #endif

Compiler Controlled Directives / conditional compilation



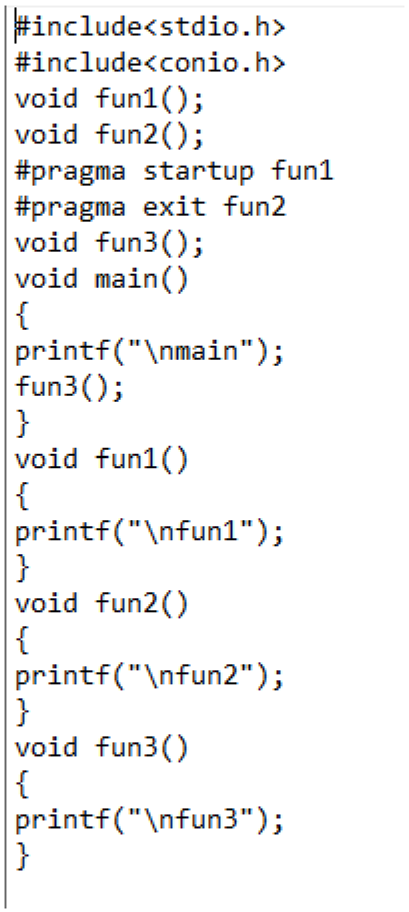
Uses of Conditional

- use different code depending on the machine, operating system
- compile same source file in two different programs
- to exclude certain code from the program but to keep it as reference for future purpose

Miscellaneous

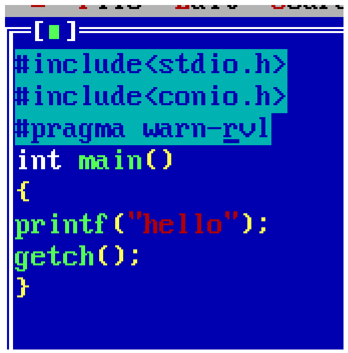
- #undef
- #pragma (specifies certain instruction)
  - #pragma startup
  - #pragma exit
  - #pragma warn
- #error (stops compilation when an error occur)
- # (Stringizing operator)
- ## (token–pasting operator)

Example of #pragma



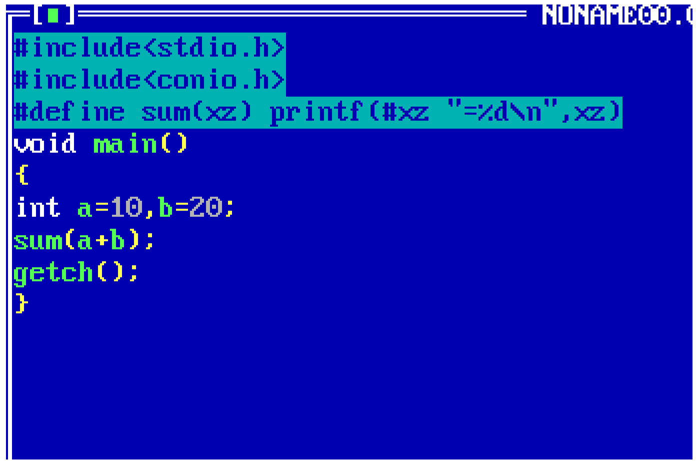
```c
#include<stdio.h>
#include<conio.h>
void fun1();
void fun2();
#pragma startup fun1
#pragma exit fun2
void fun3();
void main()
{
printf("\nmain");
fun3();
}
void fun1()
{
printf("\nfun1");
}
void fun2()
{
printf("\nfun2");
}
void fun3()
{
printf("\nfun3");
}
```

Example of #pragma warn



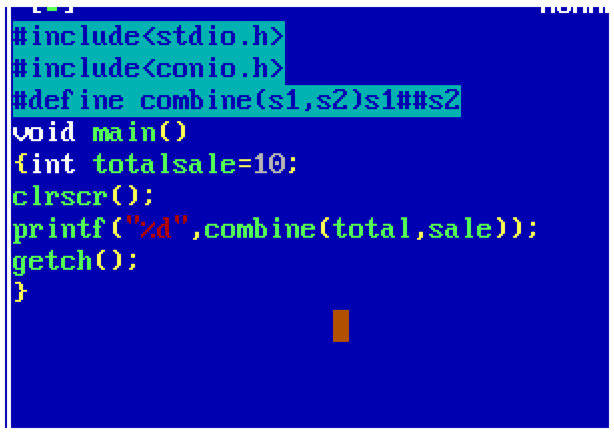
```
#include<stdio.h>
#include<conio.h>
#pragma warn-rvl
int main()
{
printf("hello");
getch();
}
```

Example of # (Stringizing operator)



```
#include<stdio.h>
#include<conio.h>
#define sum(xz) printf(#xz "=%d\n",xz)
void main()
{
int a=10,b=20;
sum(a+b);
getch();
}
```

Example of # #(token pasting)



```
#include<stdio.h>
#include<conio.h>
#define combine(s1,s2)s1##s2
void main()
{int totalsale=10;
clrscr();
printf("%d",combine(total,sale));
getch();
}
```

END