Preprocessor and Macros

C preprocessor directive

The C preprocessor is a micro processor that is used by

compiler to transform your code before compilation.

It is called micro preprocessor because it allows us to add

macros.

C Macros

A macro is a segment of code which is replaced by the value

of macro. Macro is defined by #define directive. There are

two types of macros:

Object-like Macros

Function-like Macros

Object-like Macros

The object-like macro is an identifier that is replaced by value.

It is widely used to represent numeric constants. For example:

#define PI 3.14

Here, PI is the macro name which will be replaced by the

value 3.14.

Function-like Macros

The function-like macro looks like function call. For example:

#define MIN(a,b) ((a)<(b)?(a):(b))

Here, MIN is the macro name.

C Predefined Macros

ANSI C defines many predefined macros that can be used in c

program.

Macro

Description

1

\_DATE\_

represents current date in "MMM DD YYYY"

format.

2

\_TIME\_

represents current time in "HH:MM:SS" format.

3

\_FILE\_

represents current file name.

4

\_LINE\_

represents current line number.

5

\_STDC\_

It is defined as 1 when compiler complies with

the ANSI standard.

C predefined macros example

#include <stdio.h>

void main() {

printf("File :%s\n", \_\_FILE\_\_ );

printf("Date :%s\n", \_\_DATE\_\_ );

printf("Time :%s\n", \_\_TIME\_\_ );

printf("Line :%d\n", \_\_LINE\_\_ );

printf("STDC :%d\n", \_\_STDC\_\_ );

}

C #include

The #include preprocessor directive is used to paste code of

given file into current file.

It is used include system-defined and user-defined header

files. If included file is not found, compiler renders error.

By the use of #include directive, we provide information to

the preprocessor where to look for the header files. There are

two variants to use #include directive.

#include <filename>

#include "filename"

The #include <filename> tells the compiler to look for the

directory where system header files are held.

The #include "filename" tells the compiler to look in the

current directory from where program is running.

#include directive example

#include <stdio.h>

main() {

printf("Hello C");

}

#include notes:

Note 1: In #include directive, comments are not recognized.

So in case of #include <a//b>, a//b is treated as filename.

Note 2: In #include directive, backslash is considered as

normal text not escape sequence. So in case of #include

<a\nb>, a\nb is treated as filename.

Note 3: You can use only comment after filename otherwise it

will give error.

C #define

The #define preprocessor directive is used to define constant

or micro substitution. It can use any basic data type.

Syntax:

#define token value

#include <stdio.h>

#define PI 3.14

main() {

printf("%f",PI);

}

an example of #define to create a

macro.

#include <stdio.h>

#define MIN(a,b) ((a)<(b)?(a):(b))

void main() {

printf("Minimum between 10 and 20 is: %d\n", MIN(10,20));

}

Conditional Compilation

In C programming, you can instruct preprocessor whether to

include a block of code or not. To do so, conditional directives

can be used.

It's similar to a if statement with one major difference.

The if statement is tested during the execution time to check

whether a block of code should be executed or not whereas,

the conditionals are used to include (or skip) a block of code

in your program before execution.

C #if

The #if preprocessor directive evaluates the

expression or condition.

If condition is true, it executes the code otherwise

#elseif or #else or #endif code is executed.

#include <stdio.h>

#include <conio.h>

#define NUMBER 0

void main() {

#if (NUMBER==0)

printf("Value of Number is: %d",NUMBER);

#endif

getch();

}

#include <stdio.h>

#include <conio.h>

#define NUMBER 1

void main() {

clrscr();

#if (NUMBER==0)

printf("1 Value of Number is: %d",NUMBER);

#endif

#if (NUMBER==1)

printf("2 Value of Number is: %d",NUMBER);

#endif

getch();

}

Enumeration

#include <stdio.h>

enum week{ sunday, monday, tuesday, wednesday, thursday,

friday, saturday};

int main()

{

enum week today;

today=wednesday;

printf("%d day",today+1);

return 0;

}

typedef in c

typedef is a keyword used in C language to assign alternative

names to existing datatypes.

Its mostly used with user defined datatypes, when names of

the datatypes become slightly complicated to use in

programs.

Following is the general syntax for using typedef,

typedef <existing\_name> <alias\_name>

#include<stdio.h>

#include<conio.h>

void main()

{

typedef int nccs;

nccs a,b,c;

a=10,b=20;

c=a+b;

printf("%d",c);

getch();

}

Structure definition using typedef

#include<stdio.h>

typedef struct employee

{

char name[50];

int salary;

}emp;

void main( )

{

emp e1;

printf("\nEnter Employee record:\n");

printf("\nEmployee name:\t");

scanf("%s", e1.name);

printf("\nEnter Employee salary: \t");

scanf("%d", &e1.salary);

printf("\nstudent name is %s", e1.name);

printf("\nroll is %d", e1.salary);

}

typedef and Pointers

typedef can be used to give an alias name to pointers also.

Here we have a case in which use of typedef is beneficial

during pointer declaration.

In Pointers \* binds to the right and not on the left.

int\* x, y;

By this declaration statement, we are actually declaring x as a

pointer of type int, whereas y will be declared as a

plain int variable.

typedef int\* IntPtr;

IntPtr x, y, z;

But if we use typedef like we have used in the example above,

we can declare any number of pointers in a single statement.

Memory and string handling function

1

void \*memchr(const void \*str, int c, size\_t n)

Searches for the first occurrence of the character c (an

unsigned char) in the first n bytes of the string pointed to,

by the argument str.

2

int memcmp(const void \*str1, const void \*str2, size\_t n)

Compares the first n bytes of str1 and str2.

3

void \*memcpy(void \*dest, const void \*src, size\_t n)

Copies n characters from src to dest.

4

void \*memmove(void \*dest, const void \*src, size\_t n)

Another function to copy n characters from str2 to str1.

5

void \*memset(void \*str, int c, size\_t n)

Copies the character c (an unsigned char) to the first n

characters of the string pointed to, by the argument str.

Example of memcpy

#include <stdio.h>

#include <string.h>

int main () {

char src[50] = "Nepal";

char dest[50];

strcpy(dest,"Helloooo!!");

printf("Before memcpy dest = %s\n", dest);

memcpy(dest, src, strlen(src)+1);

printf("After memcpy dest = %s\n", dest);

return(0);

}

Example of memcmp

#include <stdio.h>

#include <string.h>

int main () {

char str1[15]="nccs";

char str2[15]="NCCS";

int ret;

ret = memcmp(str1, str2, 5);

if(ret > 0) {

printf("str2 is less than str1");

} else if(ret < 0) {

printf("str1 is less than str2");

} else {

printf("str1 is equal to str2");

}

return(0);

}

Example of memchr

#include <stdio.h>

#include <string.h>

int main ()

{

char str[] = "nccs college";

char ch = 'e';

char \*ret;

ret = memchr(str, ch, strlen(str));

printf("%s", ret);

return(0);

}

Example of memset

#include <stdio.h>

#include <string.h>

int main ()

{

char str[50];

strcpy(str,"This is string.h library function");

puts(str);

memset(str,'$',7);

puts(str);

return(0);