**Macro**

# More Preprocessor features

Three special features in the C preprocessor (which is also the C++ preprocessor):

1. stringizing
2. string concatenation
3. token pasting

# Stringizing

Stringizing, introduced earlier in the book, is performed with the # directive and allows you to take an identifier and turn it into a character array.

# String concatenation

String concatenation takes place when two adjacent character arrays have no intervening punctuation, in which case they are combined.

These two features are especially useful when writing debug code.

#define DEBUG(x) cout << #x " = " << x << endl;

This prints the value of any variable.

You can also get a trace that prints out the statements as they execute:

#define TRACE(s) cerr << #s << endl; s

The #s stringizes the statement for output, and the second s reiterates the statement so it is executed.

# Token pasting

Token pasting, implemented with the ## directive, is very useful when you are manufacturing code. It allows you to take two identifiers and paste them together to automatically create a new identifier.

Example,

#define FIELD(a) char\* a##\_string; int a##\_size

class Record {

FIELD(one);

FIELD(two);

FIELD(three);

};

Each call to the FIELD( ) macro creates an identifier to hold a character array and another to hold the length of that array.

## Question:

In C, if a macro is not defined, the pre-processor assigns 0 to it by default.

After the pre-processing is over, all the undefined macros are initialized with default value 0

#include <stdio.h>

#if X == 3

    #define Y 3

#else

    #define Y 5

#endif

int main()

{

    printf("%d", Y);

    return 0;

}

Ans:

3 [🗶]

**5 [✓]**

3 or 5 depending on value of X [🗶]

Compile time error [🗶]

#include <stdio.h>

#define ISEQUAL(X, Y) X == Y

int main()

{

    #if ISEQUAL(X, 0)

        printf("Geeks");

    #else

        printf("Quiz");

    #endif

    return 0;

}

Ans:

Geeks **[✓]**

Quiz [🗶]

Any of Geeks or Quiz [🗶]

Compile time error [🗶]

Explanation:

The conditional macro #if ISEQUAL(X, 0) is expanded to #if X == 0. After the pre-processing is over, all the undefined macros are initialized with default value 0. Since macro X has not been defined, it is initialized with 0. So, Geeks is printed.

## Question:

token pasting operator ## which concatenates parameters to macro.

#include <stdio.h>

#define macro(n, a, i, m) m##a##i##n

#define MAIN macro(n, a, i, m)

int MAIN()

{

    printf("GeeksQuiz");

    return 0;

}

Compiler Error [🗶]

GeeksQuiz [✓]

MAIN [🗶]

Main [🗶]

## Question:

Pre-processor doesn't give any error if we redefine a pre-processor directive.

It may give warning though.

Pre-processor takes the most recent value before use of and put it in place of a.

#include <stdio.h>

#define a 10

int main()

{

  printf("%d ",a);

  #define a 50

  printf("%d ",a);

  return 0;

}

Compiler Error [🗶]

10 50 [✓]

50 50 [🗶]

10 10 [🗶]

## "#pragma once"

Used in a header file to avoid its inclusion more than once.

#include “stdio.h”

#include “stdio.h”

#include “stdio.h”

int main() {

printf(“Whether this statement would be printed?”);

return 0;

}

Ans:

program will not give compile error

## Question:

The pre-processing operator '#' is used to convert a string argument into a string constant.

#include <stdio.h>

#define get(s) #s

int main() {

char str[] = get(GeeksQuiz);

printf("%s", str);

return 0;

}

Compiler Error [🗶]

#GeeksQuiz [🗶]

GeeksQuiz [✓]

GGeeksQuiz [🗶]

## Question:

In C, for macros with arguments, there cannot be any space between macro name and open parenthesis.

#define INC1(a) ((a)+1)

#define INC2 (a) ((a)+1)

#define INC3( a ) (( a ) + 1)

#define INC4 ( a ) (( a ) + 1)

Ans:

Only INC1 and INC3 are correct.

Explanation:

In C, for macros with arguments, there can’t be any space between macro name and open parenthesis. That’s why only INC1 and INC3 are correct. Basically, “#define INC2 (a) ((a)+1)” results in “INC2” expansion to “(a) ((a)+1)” which is not the desired expansion.