//////////////////////////////////////////////////////////////////////////////////////////////////////////

Find\_MACRO -> fMakro

//////////////////////////////////////////////////////////////////////////////////////////////////////////

Macros and its types in C/C++

Difficulty Level : Easy

Last Updated : 12 Dec, 2022

Read

Discuss

Courses

Practice

Video

A macro is a piece of code in a program that is replaced by the value of the macro. Macro is defined by #define directive. Whenever a macro name is encountered by the compiler, it replaces the name with the definition of the macro. Macro definitions need not be terminated by a semi-colon(;).

Below are the programs to illustrate the use of macros in C/C++:

Program 1:

// C program to illustrate macros

#include <stdio.h>

// Macro definition

#define LIMIT 5

// Driver Code

int main()

{

// Print the value of macro defined

printf("The value of LIMIT"

" is %d",

LIMIT);

return 0;

}

Output

The value of LIMIT is 5

Program 2:

// C program to illustrate macros

#include <stdio.h>

// Macro definition

#define AREA(l, b) (l \* b)

// Driver Code

int main()

{

// Given lengths l1 and l2

int l1 = 10, l2 = 5, area;

// Find the area using macros

area = AREA(l1, l2);

// Print the area

printf("Area of rectangle"

" is: %d",

area);

return 0;

}

Output

Area of rectangle is: 50

Explanation:

From the above program, we can see that whenever the compiler finds AREA(l, b) in the program it replaces it with the macros definition i.e., (l\*b). The values passed to the macro template AREA(l, b) will also be replaced by the statement (l\*b). Therefore, AREA(10, 5) will be equal to 10\*5.

Types Of Macros

**Object-like Macros:** An object-like macro is a simple identifier that will be replaced by a code fragment. It is called object-like because it looks like an object in code that uses it. It is popularly used to replace a symbolic name with numerical/variable represented as constant.

Below is the illustration of a simple macro:

// C program to illustrate macros

#include <stdio.h>

// Macro definition

#define DATE 31

// Driver Code

int main()

{

// Print the message

printf("Lockdown will be extended"

" upto %d-MAY-2020",

DATE);

return 0;

}

Output

Lockdown will be extended upto 31-MAY-2020

**2. Chain Macros**: Macros inside macros are termed as chain macros. In chain macros first of all parent macro is expanded then the child macro is expanded.

Below is the illustration of a Chain Macro:

// C program to illustrate macros

#include <stdio.h>

// Macro definition

#define INSTAGRAM FOLLOWERS

#define FOLLOWERS 138

// Driver Code

int main()

{

// Print the message

printf("Geeks for Geeks have %dK"

" followers on Instagram",

INSTAGRAM);

return 0;

}

Output

Geeks for Geeks have 138K followers on Instagram

Explanation:

INSTAGRAM is expanded first to produce FOLLOWERS. Then the expanded macro is expanded to produce the outcome as 138K. This is called the chaining of macros.

**3. Multi-line Macros:** An object-like macro could have a multi-line. So to create a multi-line macro you have to use backslash-newline.

Below is the illustration of multiline macros:

// C program to illustrate macros

#include <stdio.h>

// Multi-line Macro definition

#define ELE 1, \

2, \

3

// Driver Code

int main()

{

// Array arr[] with elements

// defined in macros

int arr[] = { ELE };

// Print elements

printf("Elements of Array are:\n");

for (int i = 0; i < 3; i++) {

printf("%d ", arr[i]);

}

return 0;

}

Output

Elements of Array are:

1. 2 3

**4. Function-like Macro:** These macros are the same as a function call. It replaces the entire code instead of a function name. Pair of parentheses immediately after the macro name is necessary. If we put a space between the macro name and the parentheses in the macro definition, then the macro will not work.

A function-like macro is only lengthened if and only if its name appears with a pair of parentheses after it. If we don’t do this, the function pointer will get the address of the real function and lead to a syntax error.

Below is the illustration of function-like macros:

// C program to illustrate macros

#include <stdio.h>

// Function-like Macro definition

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

// Driver Code

int main()

{

// Given two number a and b

int a = 18;

int b = 76;

printf("Minimum value between"

" %d and %d is %d\n",

a, b, min(a, b));

return 0;

}

Output

Minimum value between 18 and 76 is 18

Here is one example that will give more understanding of Macros:

Problem: We need to find the area of a circle by defining AREA(r) Macros.

#include <iostream>

using namespace std;

#define PI 3.1416

#define AREA(r) (PI\*(r)\*(r))

int main() {

float r = 7; // radius of circle

cout<<"Area of Circle with radius " << r <<": "<< AREA(r);

return 0;

}

// This code is contributed by balbiriitbombay3602

Output

Area of Circle with radius 7: 153.938

Questions:

1. What are the advantages and disadvantages of Macros?

Ans. Macros are abbreviations for lengthy and frequently used statements. When a macro is called the entire code is substituted by a single line though the macro definition is of several lines.

The disadvantage of it is here the entire code is substituted so the program becomes lengthy if a macro is called several times.