30 January 2024

PREPROCESSOR DIRECTIVE

What is the preprocessor directive in C++?

The preprocessor is a program that is invoked by the compiler to process code before compilation. Commands for that program, known as directives, are lines of the source file beginning with the character # , which distinguishes them from lines of source program text.

#include <iostream>

using namespace std;

#define getmax(a,b) ((a)>(b)?(a):(b))

int main()

{

int x=5, y;

y = getmax(x,2);

cout<<y<<endl;

cout<<getmax(7,x)<<endl;

return 0;

}

#include <iostream>

using namespace std;

#define Num 13

int main()

{

#ifdef PI

cout<<"PI value:"<<PI<<endl;

#elif defined(Num)

cout<<"Num is:"<<Num<<endl;

#else

cout<<"Not defined"<<endl;

#endif

return 0;

}

Here are some of the most commonly used preprocessor directives in C++:

* #define:

This directive is used to define a macro. A macro is a named constant or expression that can be used throughout the program. For example, you could use a macro to define the value of PI or the size of a buffer.

* #include:

This directive is used to include the contents of another file into your program. This can be useful for including header files that contain declarations for functions and classes.

* #undef:

This directive is used to undefine a macro. This can be useful if you no longer need to use a macro or if you want to change its value.

* #ifdef:

This directive is used to check if a macro is defined. If it is, the code following the #ifdef directive will be compiled. If it is not, the code will be ignored.

* #ifndef:

This directive is used to check if a macro is not defined. If it is not, the code following the #ifndef directive will be compiled. If it is, the code will be ignored.

* #if:

This directive is used to check a condition. If the condition is true, the code following the #if directive will be compiled. If the condition is false, the code will be ignored.

* #else:

This directive is used to provide alternative code if the condition in the #if directive is false.

* #error:

This directive is used to generate an error message. If this directive is encountered, the compilation process will be stopped.

* #warning:

This directive is used to generate a warning message. If this directive is encountered, the compilation process will continue, but the warning message will be displayed.

1.Wave\_array

Sample input: 2,4,7,8,9,10

Sample output:4,2,8,7,10,9

#include <iostream>

#include <vector>

#include <algorithm>

void waveArray(std::vector<int>& arr) {

std::sort(arr.begin(), arr.end());

for (int i = 0; i < arr.size() - 1; i += 2)

std::swap(arr[i], arr[i + 1]);

}

int main() {

std::vector<int> input = {2, 4, 7, 8, 9, 10};

std::cout << "Sample input: ";

for (int num : input) std::cout << num << " ";

waveArray(input);

std::cout << "\nSample output: ";

for (int num : input) std::cout << num << " ";

return 0;

}

2.Move zeros to right

Sample output:12,0,13,0,15

Sample output:12,13,15,0,0

#include <iostream>

void moveZerosToRight(int arr[], int size) {

for (int nonZeroIndex = 0, i = 0; i < size; ++i) {

if (arr[i] != 0) {

std::swap(arr[i], arr[nonZeroIndex++]);

}

}

}

int main() {

int arr[] = {12, 0, 13, 0, 15};

int size = sizeof(arr) / sizeof(arr[0]);

std::cout << "Original array: ";

for (int i = 0; i < size; ++i) std::cout << arr[i] << " ";

std::cout << std::endl;

moveZerosToRight(arr, size);

std::cout << "After moving zeros to the right: ";

for (int i = 0; i < size; ++i) std::cout << arr[i] << " ";

std::cout << std::endl;

return 0;

}