Pre-processor in C

The preprocessor in C here is not part of the compiler, but there are separate steps in the compilation process. In the most basic way, the preprocessor in C language is the text replacement tool and the compiler instruction does not require pre-processing before being compiled.

The preprocessor in C here is not part of the compiler, but there are separate steps in the compilation process. In the most basic way, the preprocessor in C language is the text replacement tool and the compiler instruction does not require pre-processing before being compiled. We aim to preprocessor C like CPP.

All preprocessor commands start with # #. It is at least not white characters, for easy reading. Below is a list of important pre-processing cards.

Description # define Replace for preprocessor macro #include Insert a special header from another file #undef Do not define a preprocessor macro #ifdef Returns true if this macro is defined #ifndef Returns price true if this macro is not defined #if Check if the compilation condition is true #else Part replacement for #if #elif #else a #if in a #endif command End preprocessor condition #error Print error message on stderr #pragma Notice special commands to the compiler, using a standardized method

For example the preprocessor set in C

Analyze the following examples to understand the various directives.

```
#define  DO_DAI_MANG_TOI_DA  20
```

This preprocessor notifies the C compiler to replace DO_DAI_MANG_TOI_DA with 20. Using #define for constants increases the readability of the program.

```
#include #include "headercuatoi.h"
```

This preprocessor tells the compiler to retrieve the stdio.h library from **the System Library** and add the current source code. The next line tells the compiler to get the **headercuatoi.h** file from the computer directory and add the current content and source.

```
#undef  KICH_CO_FILE  #define  KICH_CO_FILE  42
```

This preprocessor tells the compiler to modify the variable KICH_CO_FILE and the new definition has the value 42.

```
#ifndef  THONGDIEP  #define  THONGDIEP  "Chao mung chang dep trai nhat nha!" #en
```

This informs the C language compiler which defines THONGDIEP if THONGDIEP is not defined.

```
#ifdef  DEBUG  /* tai day la phan lenh de debug cua ban */ #endif
```

This informs the preprocessor of the command operation if DEBUG is defined.

Macros are predefined in C

ANSI C defines a number of macros. Although each of these macros is available for you to use in the program, you should not directly edit these predefined macros.

MacroDescription__DATE__ Current date, as a character constant, in the format "MMM DD YYYY" __TIME__ Current time, as a character constant, in the format "HH: MM: SS" __FILE__ It contains the name Current file as a constant string __LINE__ It contains the current line number as a decimal constant __STDC__ Defined as 1 when the compiler compiles with ANSI standard
Try the following example:

```
#include  main () {  printf ( "File :%sn" ,  __FILE__  );  printf ( "Date :%sn"
```

When the above C program in the file **Untitled4.cpp** is compiled and executed to see the result:

Preprocessor operator in C

The C language provides the following operators to help you create macros:

The continuation operator () of the macro in C

A macro is usually included in a single line. The macro's continuation operator is often used to continue a macro if there is more than one line. For example:

```
#define  thong_diep ( a ,  b )  printf (# a  " va " #b ": nghia la Forever Alon
```

The pound sign (#) in C

The **stringize** operator **- the pound sign ('#')** , when used in a macro definition, converts a macro parameter into a string constant. This operator can be used with macros to specify a specific parameter in the parameter list. For example:

```
#include #define  thong_diep ( a ,  b )  printf (# a  " va " #b ": nghia la For
```

Compile and execute the above C program to see the results:

Token Pasting operator (##) in C

The token pasting operator (##) is used in a macro definition **that connects** two parameters. It allows two separate tokens in the macro definition to be combined into one token. Therefore it is also called a **compound operator** . For example:

```
#include #define  vidutoken ( n )  printf  ( "token" #n " = %d", token##n) int
```

Compile and execute the above C program to see the results:

How does it happen, because this example has a real output from the preprocessor:

```
printf  ( "token1 = %d" ,  token1 );
```

This example shows the concatenation of the token ## n in token34 and here we used both **stringize** and **token-pasting** .

Operator defined () in C

Preprocessor operator defined is used with constant expression to determine if an identifier is defined by #define. If the specified identifier is defined, the value is true (other than 0). If not defined, the value is false (zero). The operator defined is defined as follows:

```
#include #if !defined (THONGDIEP) #define  THONGDIEP  "Chao mung chang dep trai
```

Compile and execute the above C program to see the results:

Macro parameter in C

One of CPP's powerful features is the ability to mimic functions by using parameter macros. For example, we can have a code to square a number like this:

```
int  binhphuong ( int  x ) { return  x  *  x ; }
```

We can rewrite the above code using a macro like this:

```
#define  binhphuong ( x ) (( x ) * ( x ))
```

Macros with parameters must be defined using **#define** before they can be used. The list of parameters is enclosed in parentheses and must be immediately followed by the macro name. Spaces are not allowed between macro names and open parentheses. For example:

```
#include #define  LONNHAT ( x , y ) (( x ) > ( y ) ? ( x ) : ( y )) int  main (
```

Compile and execute the above C program to see the results:

According to Tutorialspoint

Previous article: Read - Write File in C

Next article: Header File in C