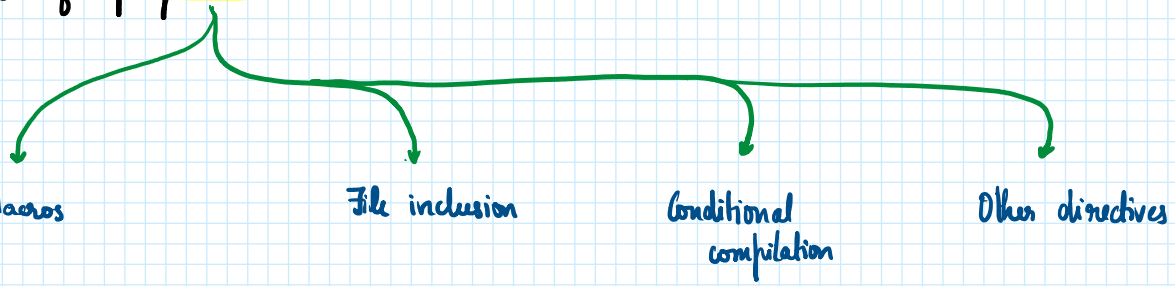
3. Preprocessor Directives, Qualifiers, Portable Program Development

14 May 2024 09:17

PREPROCESSOR DIRECTIVES

- Executed by C pre-processor
- Text substitution tool; instructs compiler to do required preprocessing before actual compilation
- Preceded by #

Types of preprocessor directives



MACROS

Piece of code in program that is given a name

- Name —(substituted during preprocessing)→ Piece of code
- # define directive used to define a macro

Eg: # define SQR(x) x*x

Features

- They do not judge anything. Eg # SQR (1+2) = 1+2*1+2 = 5
- No memory allocation
- Can define string using macros
- Macros can be used inside of macros
- Can define macro with expression / parameter
- Macros cannot be redefined with assignment operator; can be redefined with # define

CONDITIONAL COMPILATION

Certain blocks of code are compiled based on some condition

- # ifdef, # ifndef, # if, # else, # elif, # else, # endif

OTHER DIRECTIVES

#undef

Undefines an already existing macro

#pragma

QUALIFIERS

Keywords applied to a particular datatype to create a qualified datatype

Size qualifiers

Modifies the size of the datatype associated with a variable

short type <= type <= long type

Sign qualifiers

Specifies whether a variable can hold a negative value or not.

- Applies only to int, char
- ```
  unsigned int a = -10;
  printf("%u", a);
  ```

  OUTPUT: 4294967286

Type qualifiers

```
const int a = 10;
int const *p = &a; // pointer to constant integer
*p = 20;           //invalid
int b = 20;
p = &b;            //valid
int * const p = &a; // constant pointer to constant integer
p = &b;
*p = 20;           } Both invalid
```

```
volatile int a = 0;
```

- Used to prevent compiler from doing any optimisations
- Volatile values can be changed by code outside the scope of the current code
```

used to prevent compiler from doing any optimisations

- Volatile values can be changed by code outside the scope of the current code
- Also able to be changed by external hardware
- Applied in embedded systems

PORTABLE PROGRAM DEVELOPMENT

Macros to check platform

`__MINGW32__ = 1`	if	mingw on Windows
`__unix__ = 1`	if	LINUX / UNIX
`__APPLE__ = 1`	if	MacOS

} Use conditional pre-processor directives with these values to create a cross-platform program

Flushing input across platforms

Windows

```
fflush(stdin);
```

LINUX/UNIX

```
#include <stdio_ext.h>
__fpurge(stdin);
```

MacOS

```
fpurge(stdin);
```