Functions

Preprocessor

↓Preprocessor

```
#include <stdio.h>
```

C Source Code -> Preprocessor -> Compiler

allows macro

Note:

- traditionally starts in column 1 (but not necessary)
- typically occurs at the beginning of a file, however, may occur on any line inside or outside function definition

Preprocessor directives

Directive	Meaning
#include	Include the contents of a text file.
#define	Define a macro.
#undef	Cancel a previous #define.
#if	If a test succeeds, take specified action.
#ifdef	If a macro is defined, take specified actions.
#ifndef	Opposite of #ifdef--if a macro is not defined, take specified actions.
#else	If the previous #if, #ifdef, or #ifndef fails, take specified actions.
#endif	Mark the end of an #if, #ifdef, or #ifndef body.
#elif	“Else if”--a way around nested #if-#else constructs.
#line	Set line number for the compiler to use when issuing warning or error messages.
#error	Specify a compile-time error and an accompanying message.
#pragma	Provide implementation-specific information to the compiler.
#	Ignore this line.

- **File Inclusions**

#include <stdio.h>	a directory already known to the operating system.
#include “myfile.h”	to be found in the working (generally the current) directory.

#include “/home/user1/general_include/my_own_include_file”

System Header File	Purpose
ctype.h	Functions for testing and modifying characters .
float.h	Describes local floating-point conditions
limits.h	Describes local integer conditions
math.h	Math functions
setjmp.h	Nonlocal jumps
signal.h	Exception handling ;
stdarg.h	Functions with an arbitrary number of arguments .
stdio.h	input/output
stdlib.h	General utilities .
string.h	String-handling functions .
time.h	Time functions .

- **Using One File Inclusion**

  build file called all .h

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include “myfile.h”
```

So, in program we only need contain the line

```
#include “all.h”
```

If include file has been modified, all functions must be re-compiled

- **Macros**

```
#define EOF (-1)



#define NormalMeetings    3
#define SpecialMeetings   2
#define TimePerMeeting    20

#define TotalMeetings     NormalMeetings + SpecialMeetings

        /* The macro TotalMeetings expand into 3 + 2 = 5 */
        /* This is not ideal: think about TotalMeetings * <a variable>;*/
```

```
#define NormalMeetings      3
#define SpecialMeetings     2
#define TimePerMeeting      20
#define TotalMeetings       NormalMeetings + SpecialMeetings
--------------------------------
```

Macro performs simple substitution

```
#define TotalTime           TotalMeetings * TimePerMeeting
```

is equvalent to

```
#define TotalTime   NormalMeetings + SpecialMeetings* TimePerMeeting
```

/* The macro TotalTime expand into 3 + 2 * 20 = 43 instead of 100.

```
#define TotalMeetings       (NormalMeetings + SpecialMeetings)
```

Note:

```
#define TRUE 1
#define TRUE 2      /*(illegal)*/

#define TRUE 1
#undef TRUE
#define TRUE 2      /*legal*/
```

Multi-line definition

```
#define TotalMeetings \ /* carry over to next line */
            ( NormalMeetings + SpecialMeetings ) /*legal*/
```

Parameterized Macros

```
#include <stdio.h>
#include <stdlib.h>
#define PRINT5( a1, a2, a3, a4 ) \
        printf ( "\n%c\t%c\t%d\t%f", (a1), (a2), (a3), (a4) )
 /* no space between PRINT5 and '(' */
main ( ) {
        char c1 = 'C', c2 = 'Z';
        int num = 999;
        float fnum = 66.0;
        /* preprocessor substitutes PRINT5 by printf */
        PRINT5 ( c1, c2, num +1 , fnum ) ; /* space between PRINT5 &
                                              '(' is fine here */
        return EXIT_SUCCESS;
}
```

The output is

```
C       Z       1000    66.000000
```

Parameterized Macros Versus Functions

```
#include <stdio.h>
#include <stdlib.h>
#define min( x, y )  ( ( x) < (y) ) ? (x) : (y)
int max ( int x, int y );
main ( ) {
        int n1, n2;
        printf ( "\n1st num: " ); scanf ( "%d", &n1 );
        printf ( "\n2st num: " ); scanf ( "%d", &n2 );

        printf ( "\n\nMin: %d\tMax: %d\n", min (n1, n2), max ( n1, n2 ) );
        return EXIT_SUCCESS;
}
int max ( int x, int y ) {
        return ( x > y ) ? x : y;
}
```

macro function

- No arguments are passed to parameterized macro (min)
- parameterized macro returns no value
- preprocessor merely replaces macro with the definition

```
min( num1, num2 )
```

by

```
( ( (num1) < (num2) ) ? (num1) : (num2) )
```

- More efficient than function

Function calls

- have an associated overhead
- system must copies of any arguments passed to the function
  keep track of where to resume program execution when the function returns, and so on
- No type checking of arguments in macros

Advantage: above example, the macro min can find the minimum of any two numeric data types (float, int, double, etc.)

Disadvantage: can introduce errors that are difficult to uncover. Above example, the macro min also produces some value, presumably useless, when one argument is a string and the other is a float.

- Macros can produce surprising and unpredictable side effects

```
#define square( x )            (x) * (x)

        a = 3;
        b = square ( a++ );
```

above is expanded as

```
        b = (a++)*(a++);
```

implementation 1:

```
        a =4; a++ =3; a=5; a++=4; --> b=12
```

implementation 2:

```
        a++=3; a++ =3; a =4; a=5; --> b=9
```

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

```
a=3;
b=square(a++);
```

```
a ->4
b -> 9
```

Miscellaneous Directives

```
#undef TRUE
#ifdef BIG_TABLE
#define ROWS 1000
#define COLUMNS 1000
#endif
```

```
#ifndef BIG_TABLE
#define ROWS 1000
#define COLUMNS 1000
#endif
```

```
# if defined (Big_table) && !defined (small_table)
# define row 1000
# define column 1000
#define table_size (row*column)
#endif


#if defined(SUN)
#define greeting printf("welcome to sun\n")
#elif defined(IBM)
#define greeting printf("welcome to IBM\n");
…
#else
#greeting printf("welcome to no-brand name\n");
#endif
```

Exercises 5.6

- (T/F) Each macro requires a separate #define directive to create it
- Find the syntax error

```
#define PLUS
        +
```

- #define TIMES2(num) 2*num

  What is the expression TIMES2(4+5)?

  What is the expression TIMES2((4+5))?