● Preprocessor

C Source Code $\rightarrow$ Preprocessor $\rightarrow$ Compiler

Directives: commands that start with #, the language understood by the preprocessor.

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

`<...>` means a directory known to the operating system

“...” means current working directory, e.g.

`#include "mydefs.h"`

Directives may be nested, e.g:

`#include <stdio.h>`

`#include <math.h>`

`#include ”defs.h”`
We may create a file all.h containing the above and put #include “all.h” in the program.

• Macros
Macros are literally replaced by the preprocessor, e.g.
#define EOF (-1)
Example 5.6.1
Parameterized macros
Usage is similar to a function, but the implementation is different. For example,
#define PRINT3 ( e1, e2, e3 ) \nprintf ( “\ n % c \ t % c \ t % d”, (e1), \n(e2), (e3) )
Example 5.6.2
Side effects of macros

• Application: Monte Carlo integration
• Recursion

C supports recursive functions, a function that can invoke itself.

Recursive approach to solving a problem:

– Recursively decompose the problem into subproblems of the same type.
– The base cases can be solved easily.
– Combine the solutions to the subproblems to obtain a solution to the original problem.

Pay attention to the base bases to avoid infinite recursion!

Example 5.8.1
Example 5.8.2
Example 5.8.3
Fibonacci sequence: 1, 2, 3, 5, 8, 13, ...
Efficiency: recursive vs. iterative

Example 5.8.4

Tail recursive function: the recursive call occurs in the last statement of the function. Translate tail recursive to iterative.

• Application: Recursive tiling
ARRAYS

• An array contains a sequence of variables of the same type.
More convenience than using multiple individual variables.
Define an array in C:
float temps[100];
Example 6.1.1

• Array indexes and offsets
Index starts with 0.
C does not check the length of the array.
Array length must be known at compile time, e.g.:
int array[10]; int array2[ MAX + 1];
Index may not be known until run time.
• Initialization
  An array can be initialized in the definition.
  Examples 6.2.2, 6.2.3 and 6.2.4.

• Arrays and pointers
  Two ways to characterize an array’s name:
  as a pointer to the first cell in the array.
  as an expression whose value is the address of the first cell in the array.
  Example 6.2.5.
  Array’s name is a pointer constant (can not be changed)
  Pointer variables (more later), Example 6.2.6.

• Implementation issues
  C programmer can access an element in an array without knowing the size of the element, e.g. int.
• Use of sizeof operator
  Example 6.2.7

• Application: The Fourier Transform

• Character strings
  C represents a string as an array of characters with each cell occupying one cell in the array and ending with null character ‘\ 0’.

• Arrays as function arguments
  Provide the array name and the last element’s position in the function call.
  Example 6.5.1.
• String-handling functions

C does not have built-in operators for string handling.
Use standard library or user’s own functions.

– String concatenation

The concatenation of two strings, $s_1$ and $s_2$ is the first string $s_1$ followed by the second string $s_2$.

Function strcat:
events two string argument and stores the result in the first string.

Example 6.6.1

Function strncat:
Similar to strcat except that there is a third argument, which specifies the maximum number of characters to be includ-
ed from the second string.

Example 6.6.2

– String comparison

Comparison result of two different strings:

(1) The strings have different lengths and each character in the shorter is identical to the corresponding character in the longer string.

(2) The strings have the same or different lengths and at some position the characters in the strings differ.

How to determine the order of two strings?

Lexicographic order:

In (1), shorter string precedes the longer one.

In (2), suppose the leftmost position is $p$ in the string at which the characters
differ. The order is defined by the encoding of the characters (e.g. ASCII) at position \( p \).

Figure 6.13.

Function `strcmpl`

expects two arguments, the addresses of null-terminated character strings. It compares them and returns

* 0 if the two are equal.
* A negative integer if the first one is lexicographically less than the second.
* A positive integer if the first one is lexicographically greater than the second.

Example 6.6.3

Function `strncmpl`

Similar to `strcmpl` except that there is a
third argument, which specifies the maximum number of characters to be used in the comparison.

Example 6.6.5

- String copy

Copy all or part of the second string into the first string.

Function strcpy:

Example 6.6.5

Function strncpy:

strncpy( strings1, string2, max_len);

max_len specifies the number of characters to copy.

Note that when using strncpy, the resulting string is not null terminated if the number of nonnull characters in string2 is greater than or equal to max_len.
Example 6.6.6

- String length
  \texttt{strlen}: returns the number of nonnull characters up to the null terminator.

Example 6.6.7.

- Substring/character search
  \texttt{strstr}: search for a substring
  \texttt{strchr}: search for the first occurrence of a character
  \texttt{strchr}: search for the last occurrence of a character

Example 6.6.8

- Application: computing a string’s length
• Multidimensional arrays

  – Dimension: the number of indexes used to access a particular element in an array.
  A multidimensional array sometimes is more natural and convenient than a one-dimensional array.
  All multidimensional arrays are still implemented as one-dimensional array.
  Example 6.8.1

  – Initialization:
  Can be done in the definition.
  Example 6.8.3.

  – As function arguments
  Must specify the number of cells in all dimensions beyond the first in the declaration.
Example 6.8.4

- **Application:** matrix multiplication
- **Application:** solving a linear system of equations
- **Application:** sorting and searching