【C】 Day4(2)

⚪ Course	Advanced C
🗐 Study Date	@April 4, 2022

【Ch7】 Input and Output

7.1 Standard Input and Output

The simplest input mechanism is to read one character at a time from the standard input, normally the keyboard, with `getchar`

```
int getchar(void)
```

`getchar` returns the next input character each time it is called, or `EOF` when it encounters end of file. The symbolic constant EOF is defined in `<stdio.h>`. The value is typically -1.

In many environments, a file may be substituted for the keyboard by using the `<` convention for input redirection: if a program prog uses `getchar`, then the command line

```
prog < infile
```

causes prog to read characters from `infile` instead.

A program to print out the given file:

```
#include <stdio.h>

int main() {
  char buffer;
  while((buffer = getchar()) != EOF)
    printf("%c", buffer);
  return 0;
}
```

If we type

```
prompt> .\a.out < file.txt
```

The program will print out the contents of the input file.

The function

```
int putchar(int)
```

is used for output: `putchar(c)` puts the character c on the standard output, which is by default the screen.

`putchar` returns the character written, or EOF if an error occurs. Again, output can usually be directed to a file with `> filename` : if prog uses `putchar`

```
prog > outfile
```

Each source file that refers to an input/output library function must contain the line

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

When the name is bracketed by `<` and `>` a search is made for the header in a standard set of places(for example, on UNIX systems, typically in the directory `/usr/include`)

For example, consider the program `lower` , which converts its input to lower case:

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

int main() {
  int c;

  while((c = getchar()) != EOF)
    putchar(tolower(c));
  return 0;
}
```

"Functions" like `getchar` and `putchar` in `<stdio.h>` and `tolower` in `<ctype.h>` are often macros, thus avoiding the overhead of a function call per character.