I/O Issues in C

CHEN Wang

CSCI2100 Data Structures Tutorial 3

Content

- **Introduction**
- **Input/Output Functions**
  - **printf()**
  - **scanf()**
  - **getchar() and putchar()**
  - **gets() and fgets()**
- **Constants**
  - **Constants**
  - **#define**
  - **String Literal**

Content

- **Introduction**
- **Input/Output Functions**
  - **printf()**
  - **scanf()**
  - **getchar() and putchar()**
  - **gets() and fgets()**
- **Constants**
  - **Constants**
  - **#define**
  - **String Literal**

How to read the input data and write the output data?

Exercise 1.24 *You are asked to write a C program to reverse a sentence including all punctuation.*

Input *The input consists of the number of test cases, m, in the first line and followed by m test cases.*

Each test case consists of a string with less than 256 characters

An example is as follows,

```
3
.semit 5 noitartsiger esruoc ym deliaf evah I esuaceb ,SISUC etah I
.teertS rekaB B122 ta talf a erahs ot etamtalf a rof gnikool si semloH kcolrehS
.0102 yluJ 52 no srallod 00.000,000,5 now I
```

Output *The output should be m lines of reversed sentence. Each line with one reversed sentence.*

```
I hate CUSIS, because I have failed my course registration 5 times.
Sherlock Holmes is looking for a flatmate to share a flat at 221B Baker Street.
I won 5,000,000.00 dollars on 25 July 2010.
```

Hint *You can use "scanf()" to read the number of test cases and "gets()" to read an input string.*

What’s on PC2 System? An Example

124.in

```
3
emit 5 noitartsiger esruoc ym deliaf evah I esuaceb ,SISUC etah I
.teertS rekaB B122 ta talf a erahs ot etamtalf a rof gnikool si semloH kcolrehS
.0102 yluJ 52 no srallod 00.000,000,5 now I

~
~
```

124.out

```
I hate CUSIS, because I have failed my course registration 5 times.
Sherlock Holmes is looking for a flatmate to share a flat at 221B Baker Street.
I won 5,000,000.00 dollars on 25 July 2010.
```

Output format:

- If the last valid character of a string s is not ‘\n’:

  printf(“%s\n”, s);

- Else:

  printf(“%s”, s);

C program skeleton

- In short, the basic skeleton of a C program looks like this:

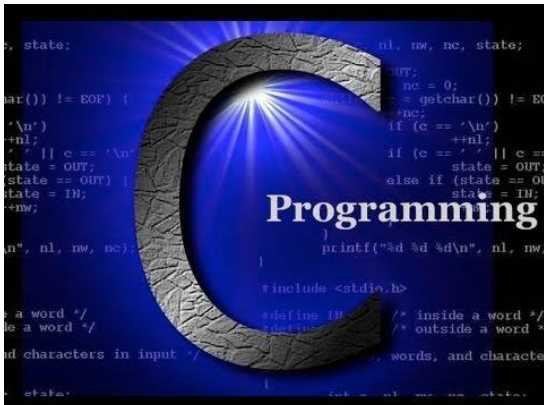
```
#include <stdio.h>
int main(void)
{
        statement(s);
        return(0);
}
```

Preprocessor directives

Function main

Start of segment

End of segment



Input/Output Operations

- **Input operation**
  - an instruction that copies data from an input device into memory

- **Output operation**
  - an instruction that displays information stored in memory to the output devices (such as the monitor screen)

Input/Output Functions

- A C function that performs an input or output operation
- A few functions that are pre-defined in the header file **<stdio.h>** such as :
  - ***printf()***
  - ***scanf()***
  - ***getchar()*** & ***putchar()***
  - ***gets()*** & ***fgets()***

Content

- Introduction
- **Input/Output Functions**
  - **printf()**
  - scanf()
  - getchar() and putchar()
  - gets() and fgets()
- Constants
  - Constants
  - #define
  - String Literal

The *printf()* function

- Used to send data to the standard output (usually the monitor) to be printed according to specific format.
- General format:
  - `printf("string literal");`
    - A sequence of any number of characters surrounded by double quotation marks.
  - `printf("format string", variables);`
    - Format string is a combination of **text**, **conversion specifier** and **escape sequence**.

The *printf()* function cont…

- Example:
  - `printf("Thank you\n");`

```
Thank you
```

  - `printf ("Total sum is: %d\n", sum);`

```
Total sum is: 50
```

Assuming that the value of sum is 50

- **%d** is a **placeholder** (conversion specifier)
  - marks the display position for a type integer variable
  - Common Conversion Identifier used in printf function
- **\n** is an **escape sequence**
  - moves the cursor to the new line

	printf
int	%d
float	%f
double	%f
char	%c
string	%s

Placeholder/Conversion Specifier

No	Conversion Specifier	Output Type	Output Example
1	%d	Signed decimal integer	76
2	%i	Signed decimal integer	76
3	%o	Unsigned octal integer	134
4	%u	Unsigned decimal integer	76
5	%x	Unsigned hexadecimal (small letter)	9c
6	%X	Unsigned hexadecimal (capital letter)	9C
7	%f	Integer including decimal point	76.0000
8	%e	Signed floating point (using e notation)	7.6000e+01
9	%E	Signed floating point (using E notation)	7.6000E+01
10	%g	The shorter between %f and %e	76
11	%G	The shorter between %f and %E	76
12	%c	Character	‘7’
13	%s	String	‘76'

Escape Sequence

Escape Sequence	Effect
\a	Beep sound
\b	Backspace
\f	Formfeed (for printing)
\n	New line
\r	Carriage return
\t	Tab
\v	Vertical tab
\\	Backslash
\”	“ sign
\o	Octal decimal
\x	Hexadecimal
\0	NULL

Content

- Introduction
- **Input/Output Functions**
  - printf()
  - **scanf()**
  - getchar() and putchar()
  - gets() and fgets()
- Constants
  - Constants
  - #define
  - String Literal

The scanf() function

- Read data from the standard input device (usually keyboard) and store it in a variable.
- General format:
  - `scanf("format string", &variable);`
- Notice ampersand (&) operator :

8280	50
address	variable

  - C address of operator
  - it passes the address of the variable instead of the variable itself
  - tells the scanf() where to find the variable to store the new value
- Format string is a combination of conversion specifier and escape sequence (if any).

The scanf() function cont…

- Common Conversion Identifier used in printf and scanf functions.

	printf	scanf
int	%d	%d
float	%f	%f
double	%f	%lf
char	%c	%c
string	%s	%s

- Example :

```
int age;
printf("Enter your age:");
scanf("%d", &age);
```

The scanf() function cont…

- If you want the user to enter more than one value, you serialize the inputs.

- Example:

```
float height, weight;

printf("Please enter your height and weight:");
scanf("%f%f", &height, &weight);
```

Content

- Introduction
- **Input/Output Functions**
  - printf()
  - scanf()
  - **getchar() and putchar()**
  - gets() and fgets()
- Constants
  - Constants
  - #define
  - String Literal

getchar() and putchar()

- ***getchar()*** - read ***one*** character from standard input
- ***putchar()*** - write ***one*** character to standard output
- Example:

```
Please type a character: h
You have typed this character: h
```

```
#include <stdio.h>
int main(void)
{
  char my_char;
  printf("Please type a character: ");
  my_char = getchar();
  printf("You have typed this character: ");
  putchar(my_char);
  return (0);
}
```

getchar() and putchar() cont

- Alternatively, you can write the previous code using normal ***printf*** / ***scanf*** and %c placeholder.
- Example:

```
Please type a character: h
You have typed this character: h
```

```
#include <stdio.h>
int main(void)
{
  char my_char;
  printf("Please type a character: ");
  scanf("%c",&my_char);
  printf("You have typed this character: %c", my_char);
  return(0);
}
```

Content

- Introduction
- **Input/Output Functions**
  - printf()
  - scanf()
  - getchar() and putchar()
  - **gets() and fgets()**
- Constants
  - Constants
  - #define
  - String Literal

gets() and fgets()

- ***gets()*** - read ***one*** string from standard input.
  - It stops when either the newline character is read or when the end-of-file is reached, whichever comes first.
- ***fgets()*** - read ***one*** string from the specified stream.
  - It stops when either (n-1) characters are read where n is the maximum number of characters to be read (including the final null-character) , the newline character is read, or the end-of-file is reached, whichever comes first.

```
#include <stdio.h>
int main(void)
{
  char s[256];
  printf("Please type a string within 255 characters: \n");
  gets(s);
  printf("%s\n", s);
  printf("The above is the string you typed");
  return (0);
}
```

```
Please type a string within 255 characters:
hello
hello
The above is the string you typed
Process returned 0 (0x0)   execution time : 4.892 s
Press any key to continue.
```

gets() and fgets()

- ***gets()*** - read ***one*** string from standard input.
  - It stops when either the newline character is read or when the end-of-file is reached, whichever comes first.
- ***fgets()*** - read ***one*** string from the specified stream.
  - It stops when either (n-1) characters are read where n is the maximum number of characters to be read (including the final null-character) , the newline character is read, or the end-of-file is reached, whichever comes first.

```
#include <stdio.h>
int main(void)
{
  char s[256];
  printf("Please type a string within 255 characters: \n");
  gets(s);
  printf("%s\n", s);
  printf("The above is the string you typed");
  return (0);
}
```

```
Please type a string within 255 characters:
hello
hello
The above is the string you typed
Process returned 0 (0x0)   execution time : 4.892 s
Press any key to continue.
```

gets() and fgets()

- ***gets()*** - read ***one*** string from standard input.
  - It stops when either the newline character is read or when the end-of-file is reached, whichever comes first.
- ***fgets()*** - read ***one*** string from the specified stream.
  - It stops when either (n-1) characters are read where n is the maximum number of characters to be read (including the final null-character) , the newline character is read, or the end-of-file is reached, whichever comes first.

```
#include <stdio.h>
int main(void)
{
  char s[256];
  printf("Please type a string within 255 characters: \n");
  fgets(s, 256, stdin);
  printf("%s", s);
  printf("The above is the string you typed");
  return (0);
}
```

```
Please type a string within 255 characters:
hello
hello
The above is the string you typed
Process returned 0 (0x0)   execution time : 4.892 s
Press any key to continue.
```

gets() and fgets()

- ***gets()*** - read ***one*** string from standard input.
  - It stops when either the newline character is read or when the end-of-file is reached, whichever comes first.
- ***fgets()*** - read ***one*** string from the specified stream.
  - It stops when either (n-1) characters are read where n is the maximum number of characters to be read (including the final null-character) , the newline character is read, or the end-of-file is reached, whichever comes first.

```
#include <stdio.h>
int main(void)
{
  char s[256];
  printf("Please type a string within 255 characters: \n");
  fgets(s, 256, stdin);
  printf("%s", s);
  printf("The above is the string you typed");
  return (0);
}
```

```
Please type a string within 255 characters:
hello
hello
The above is the string you typed
Process returned 0 (0x0)   execution time : 4.892 s
Press any key to continue.
```

gets() will not store "\n"
fgets stores "\n"

gets() after scanf()

```
#include <stdio.h>
int main(void)
{ int n;
  char s[256];
  printf("Please type an integer for one line and a string for
the other line:\n");
  scanf("%d", &n);
  gets(s);
  printf("You typed:\n");
  printf("%d\n", n);
  printf("%s", s);
return (0);
}
```

```
Please type an integer for one line and a string for the other line:
20
You typed:
20

Process returned 0 (0x0)   execution time : 2.769 s
Press any key to continue.
```

start: 20\n end: 20\n n:20

start: 20\n end: 20\n s:['\0', '\0', …, '\0']

'\n' is inserted when you press the "Enter" key.

gets() will not store "\n"
fgets stores "\n"

gets() after scanf()

```
#include <stdio.h>
int main(void)
{ int n;
  char s[256];
  printf("Please type an integer for one line and a string for the
other line:\n");
  scanf("%d\n", &n);
  gets(s);
  printf("You typed:\n");
  printf("%d\n", n);
  printf("%s", s);
return (0);
}
```

```
Please type an integer for one line and a string for the other line:
20
abc
You typed:
20
abc
Process returned 0 (0x0)   execution time : 5.456 s
Press any key to continue.
```

start: 20\n

end: 20\n

n:20

start: 20\n
abc\n

end: 20\n
abc\n

s:['a', 'b', 'c', '\0', …]

'\n' is inserted when you press the "Enter" key.

gets() will not store "\n"
fgets stores "\n"

fgets() after scanf()

```
#include <stdio.h>
int main(void)
{ int n;
  char s[256];
  printf("Please type an integer for one line and a string for
the other line:\n");
  scanf("%d", &n);
  fgets(s,256,stdin);
  printf("You typed:\n");
  printf("%d\n", n);
  printf("%s", s);
return (0);
}
```

```
Please type an integer for one line and a string for the other line:
20
You typed:
20


Process returned 0 (0x0)   execution time : 1.852 s
Press any key to continue.
```

start: 20\n end: 20\n n:20

start: 20\n end: 20\n s:['\n', '\0', …, '\0']

'\n' is inserted when you press the "Enter" key.

gets() will not store "\n"
fgets stores "\n"

fgets() after scanf()

```
Please type an integer for one line and a string for the other line:
20
abc
You typed:
20
abc

Process returned 0 (0x0)   execution time : 6.115 s
Press any key to continue.
```

```
#include <stdio.h>
int main(void)
{ int n;
  char s[256];
  printf("Please type an integer for one line and a string for the other line:\n");
  scanf("%d\n", &n);
  fgets(s,256,stdin);
  printf("You typed:\n");
  printf("%d\n", n);
  printf("%s", s);
return (0);
}
```

start: 20\n

end: 20\n

n:20

start: 20\n
abc\n

end: 20\n
abc\n

s:['a', 'b', 'c', '\n', '\0'…]

'\n' is inserted when you press the "Enter" key.

gets() will not store "\n"
fgets stores "\n"

Content

- Introduction
- Input/Output Functions
  - printf()
  - scanf()
  - getchar() and putchar()
  - gets() and fgets()
- Constants
  - Constants
  - #define
  - String Literal

Constants

- **Character constants**
  - A character enclosed in a single quotation mark
  - Example:
    - `const char letter = 'n';`
    - `const char number = '1';`
    - `printf("%c", 'S');`
- **Enumeration**
  - Values are given as a list
  - Example:
    - `enum Days { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };`

Constant example – volume of a cone

```
#include <stdio.h>

int main(void)
{
   const double pi = 3.412;
   double height, radius, base, volume;

   printf("Enter the height and radius of the cone:");
   scanf("%lf %lf", &height, &radius);

   base = pi * radius * radius;
   volume = (1.0/3.0) * base * height;

   printf("The volume of a cone is %f ", volume);
   return (0);
}
```

Content

- Introduction
- Input/Output Functions
  - printf()
  - scanf()
  - getchar() and putchar()
  - gets() and fgets()
- Constants
  - Constants
  - **#define**
  - String Literal

#define

```
#include <stdio.h>
#define pi 3.142

int main(void)
{
  double height, radius, base, volume;

  printf("Enter the height and radius of the
  cone:");
  scanf("%lf %lf", &height, &radius);

  base = pi * radius * radius;
  volume = (1.0/3.0) * base * height;

  printf("The volume of a cone is %f ", volume);
  return (0);
}
```

Content

- Introduction
- Input/Output Functions
  - printf()
  - scanf()
  - getchar() and putchar()
  - gets() and fgets()
- Constants
  - Constants
  - #define
  - **String Literal**

String Literal

- A sequence of any number of characters surrounded by double quotation marks “ ”.
- Example of usage in C program:

```
printf(“What a beautiful day.\n”);
```

```
What a beautiful day.
```

- To have double quotation marks as part of the sentence, precede the quote with backslash

```
printf(“He shouted \“stop!\” to the thief.\n”);
```

```
He shouted “stop!” to the thief.
```

Thanks!

Q&A