ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

C PROGRAMMING INTRODUCTION

WEEK 3: STANDARD OUTPUT INTRODUCTION

Topic of this week

- Output
  - Class Lecture Review
    - Presentation of results
    - **printf**
    - Streams
      - puts, putchar (in <stdio.h>)
  - Programming Exercises

Input/Output in C

- C has no built-in statements for input or output.
- A library of functions is supplied to perform these operations. The I/O library functions are listed the “header” file <stdio.h>.
- You do not need to memorize them, just be familiar with them.

Streams

- Streams
  - Sequences of characters organized into lines
    - ends with new line character
    - ANSI C must support lines of at least 254 characters
  - Performs all input and output
  - Can often be redirected
    - Standard input - keyboard
    - Standard output - screen
    - Standard error - screen

Formatting Output with `printf`

- **`printf`**
  - **`precise output formatting`**
    - Conversion specifications: flags, field widths, precisions, etc.
  - Can perform rounding, aligning columns, right/left justification, inserting literal characters, exponential format, hexadecimal format, and fixed width and precision
- Format

  `printf(` format-control-string**,** other-arguments **`);`**
  - format control string: includes a listing of the data types of the variables to be output and, optionally, some text and control character(s).
  - other-arguments: correspond to each conversion specification in format-control-string
    - each specification begins with a percent sign, ends with conversion specifier

Printing Integers

- Integer
  - Whole number (no decimal point): `25, 0, -9`
  - Positive, negative, or zero
  - Only minus sign prints by default (later we shall change this)

Conversion Specifier	Description
d	Display a signed decimal integer.
i	Display a signed decimal integer. (*Note:* The **i** and **d** specifiers are different when used with **scanf**.)
o	Display an unsigned octal integer.
u	Display an unsigned decimal integer.
x or X	Display an unsigned hexadecimal integer. **X** causes the digits **0-9** and the letters **A-F** to be displayed and **x** causes the digits **0-9** and **a-f** to be displayed.
h or **l** (letter **l**)	Place before any integer conversion specifier to indicate that a **short** or **long** integer is displayed respectively. Letters **h** and **l** are more precisely called *length modifiers*.

Example 1

```
1 #include <stdio.h>
2
3 int main()
4 {
5         printf( "%d\n", 455 );
6         printf( "%i\n", 455 );/*i same as d*/
7         printf( "%d\n", +455 );
8         printf( "%d\n", -455 );
9         printf( "%hd\n", 32000 );
10        printf( "%ld\n", 2000000000 );
11        printf( "%o\n", 455 );
12        printf( "%u\n", 455 );
13        printf( "%u\n", -455 );
14        printf( "%x\n", 455 );
15        printf( "%X\n", 455 );
16
17        return 0;
18    }
```

```
455
455
455
-455
32000
2000000000
707
455
65081
1c7
1C7
```

Printing Floating-Point Numbers

- Floating Point Numbers
  - Have a decimal point (`33.5`)
  - Exponential notation (computer's version of scientific notation)
    - `150.3` is `1.503 x 10²` in scientific
    - `150.3` is `1.503E+02` in exponential (`E` stands for exponent)
    - use `e` or `E`
  - `f` - print floating point with at least one digit to left of decimal
  - `g` (or `G`) - prints in `f` or `e`(`E`) with no trailing zeros (`1.2300` becomes `1.23`)
    - Use exponential if exponent less than `-4`, or greater than or equal to precision (6 digits by default)

Example 2

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf( "%e\n", 1234567.89 );
6     printf( "%e\n", +1234567.89 );
7     printf( "%e\n", -1234567.89 );
8     printf( "%E\n", 1234567.89 );
9     printf( "%f\n", 1234567.89 );
10    printf( "%g\n", 1234567.89 );
11    printf( "%G\n", 1234567.89 );
12
13    return 0;
14     }
```

```
1.234568e+006
1.234568e+006
-1.234568e+006
1.234568E+006
1234567.890000
1.23457e+006
1.23457E+006
```

Printing Strings and Characters

- **`c`**
  - Prints **`char`** argument
  - Cannot be used to print the first character of a string
- **`s`**
  - Requires a pointer to **`char`** as an argument
  - Prints characters until **`NULL`** (`'\0'`) encountered
  - Cannot print a **`char`** argument
- Remember
  - Single quotes for character constants (**`'z'`**)
  - Double quotes for strings **`"z"`** (which actually contains two characters, **`'z'`** and `'\0'`)

Example 3

```
1  #include <stdio.h>
2
3  int main()
4  {
5     char character = 'A';
6     char string[] = "This is a string";
7     const char *stringPtr = "This is also a string";
8
9     printf( "%c\n", character );
10    printf( "%s\n", "This is a string" );
11    printf( "%s\n", string );
12    printf( "%s\n", stringPtr );
13
14    return 0;
15 }
```

```
A
This is a string
This is a string
This is also a string
```

Other Conversion Specifiers

- **`p`**
  - Displays pointer value (address)
- **`n`**
  - Stores number of characters already output by current **`printf`** statement
  - Takes a pointer to an integer as an argument
  - Nothing printed by a **`%n`** specification
  - Every **`printf`** call returns a value
    - Number of characters output
    - Negative number if error occurs
- **`%`**
  - Prints a percent sign
  - **`%%`**

Example 4

```
1#include <stdio.h>
2
3int main()
4{
5    int *ptr;
6    int x = 12345, y;
7
8    ptr = &x;
9    printf( "The value of ptr is %p\n", ptr );
10    printf( "The address of x is %p\n\n", &x );
11
12    printf("Total characters printed on this line is:%n",&y );
13    printf( " %d\n\n", y );
14
15    y = printf( "This line has 28 characters\n" );
16    printf( "%d characters were printed\n\n", y );
17
18    printf( "Printing a %% in a format control string\n" );
19
20    return 0;
21}
```

```
The value of ptr is 0065FDF0
The address of x is 0065FDF0

Total characters printed on this line is: 41

This line has 28 characters
28 characters were printed

Printing a % in a format control string
```

Printing with Field Widths and Precisions

- Field width
  - Size of field in which data is printed
  - If width larger than data, default right justified
    - If field width too small, increases to fit data
    - Minus sign uses one character position in field
  - Integer width inserted between `%` and conversion specifier
  - **`%4d`** - field width of 4

Printing with Field Widths and Precisions (II)

- Precision
  - Meaning varies depending on data type
  - Integers (default 1) - minimum number of digits to print
    - If data too small, prefixed with zeros
  - Floating point - number of digits to appear after decimal (`e` and `f`)
    - For `g` - maximum number of significant digits
  - Strings - maximum number of characters to be written from string

Printing with Field Widths and Precisions (III)

- Format
  - Precision: use a dot (.) then precision number after %

    `%.3f`
  - Can be combined with field width

    `%5.3f`
  - Can use integer expressions to determine field width and precision
    - Use *
    - Negative field width - left justified
    - Positive field width - right justified
    - Precision must be positive


```
printf( "%*.*f", 7, 2, 98.736 );
```

Example 5

```
1#include <stdio.h>
2
3 int main()
4{
5    int i = 873;
6    double f = 123.94536;
7   char s[] = "Happy Birthday";
8
9    printf( "Using precision for integers\n" );
10    printf( "\t%.4d\n\t%.9d\n\n", i, i );
11    printf( "Using precision for floating-point numbers\n" );
12    printf( "\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f );
13    printf( "Using precision for strings\n" );
14    printf( "\t%.11s\n", s );
15
16    return 0;
17}
```

```
Using precision for integers
        0873
        000000873

Using precision for floating-point numbers
        123.945
        1.239e+02
        124

Using precision for strings
        Happy Birth
```

Using Flags in the `printf` Format-Control String

- Flags
  - Supplement formatting capabilities
  - Place flag immediately to the right of percent sign
  - Several flags may be combined

Flag	Description
- (minus sign)	Left-justify the output within the specified field.
+ (plus sign)	Display a plus sign preceding positive values and a minus sign preceding negative values.
space	Print a space before a positive value not printed with the **+** flag.
#	Prefix **0** to the output value when used with the octal conversion specifier **o**.
	Prefix **0x** or **0X** to the output value when used with the hexadecimal conversion specifiers **x** or **X**.
	Force a decimal point for a floating-point number printed with **e**, **E**, **f**, **g** or **G** that does not contain a fractional part. (Normally the decimal point is only printed if a digit follows it.) For **g** and **G** specifiers, trailing zeros are not eliminated.
0 (zero)	Pad a field with leading zeros.

Example 6

```
1#include <stdio.h>
2
3 int main()
4 {
5   printf( "%10s%10d%10c%10f\n\n", "hello", 7, 'a', 1.23 );
6  printf( "%-10s%-10d%-10c%-10f\n", "hello", 7, 'a', 1.23 );
7   return 0;
8}
```

```
hello         7         a  1.230000

hello     7         a         1.230000
```

Example 7

```
1  #include <stdio.h>
2
3  int main()
4  {
5          int c = 1427;
6          double p = 1427.0;
7
8          printf( "%#o\n", c );
9          printf( "%#x\n", c );
10         printf( "%#X\n", c );
11         printf( "\n%g\n", p );
12         printf( "%#g\n", p );
13
14         return 0;
15 }
```

```
02623
0x593
0X593

1427
1427.00
```

Printing Literals and Escape Sequences

- Printing Literals
  - Most characters can be printed
  - Certain "problem" characters, such as the quotation mark "
  - Must be represented by escape sequences
    - Represented by a backslash \ followed by an escape character

Printing Literals and Escape Sequences (II)

Escape sequence	Description
`\'`	Output the single quote (') character.
`\"`	Output the double quote (") character.
`\?`	Output the question mark (?) character.
`\\`	Output the backslash (\) character.
`\a`	Cause an audible (bell) or visual alert.
`\b`	Move the cursor back one position on the current line.
`\f`	Move the cursor to the start of the next logical page.
`\n`	Move the cursor to the beginning of the next line.
`\r`	Move the cursor to the beginning of the current line.
`\t`	Move the cursor to the next horizontal tab position.
`\v`	Move the cursor to the next vertical tab position.

Exercises 3.1

- Write a program that shows the size of basic data types, such as: int, long short, double, char…
- You can use **sizeof** function to perform this task.
- e.g: sizeof(int);

Solution

```
#include <stdio.h>

main()
{

  printf("      THE SIZE OF BASIC DATA TYPES\n\n");

  printf("int  %d\n",sizeof(int));
  printf("short int  %d\n",sizeof(short int));
  printf("long int  %d\n",sizeof(long int));
  printf("unsigned int  %d\n",sizeof(unsigned int));
  printf("unsigned short  %d\n",sizeof(unsigned short));
  printf("unsigned long  %d\n",sizeof(unsigned long));

}
```

Exercises 3.2

- Write the following program. Compile, link and run it.

```
#include<stdio.h>

void main ()
{
  int year;
  float height;

  year = 21;
  height = 1.77;

  printf("Ali is %d years old and %f meter height\n", year, height);
}
```

Exercises 3.3

- Write a program that asks your name and then greets you.
- You can use scanf() function to read data with specified format from keyboard.
- E.g:

```
char word[20];
scanf("%19s", word);
```

Solution

```
#include <stdio.h>

int main(void) {
   char name[16];  /* string to hold name */

   printf("What's your name? ");
   scanf("%15s", name);
   printf("Hi there, %s!\n", name);
   return 0;
}
```

Exercises 3.4

- Now it's time for you to do some programming of your own. We want you to write a C program that will read in two integers n and m and print out the sum of all the values between n and m inclusive. The program should look like this when it's working:

```
Enter first number: 3
Enter second number: 5
Sum 3+5 = 8
```

Solution

```
#include <stdio.h>

int main(void) {
    int n, m; /* lower and upper bounds */
    int sum;  /* accumulated sum */

    /*
     * Get the numbers
     */
    printf("Enter first number: ");
    scanf("%d", &n);
    printf("Enter second number: ");
    scanf("%d", &m);
```

Solution

```
/*
 * Compute sum of n and m
 * (also, display inputs for user to check)
 */
sum = n+m;


/*
 * Print results
 */
printf("Sum of %d and %d = %d\n", n, m, sum);

return 0;
```

Exercise 3.5

- The BK library™ DVD shop has three rental rates

Type of rent	Rent per disk
Overnight	$7.00
Three-day	$5.00
Weekly	$3.00

- Write a simple C program to input the day of the week, and the number of overnight, three-day and weekly DVDs the customer is renting. Compile this program, and print out the input values to ensure that they are read correctly.
- Update your program to compute the total cost of renting the DVDs

Hint

- Note: since the day of the week is indicated by a single character, you will need to define a set of characters, e.g., ‘m’ for Monday, ‘t’ for Tuesday, and ‘h’ for Thursday.
- When reading a single character, use scanf(“ %c”,&day) to skip leading blanks.

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!

soict.hust.edu.vn/ fb.com/groups/soict