Data input/output

Data input/ouput

- To read and write data in C, we use two standard functions that include in the file <stdio.h>

- printf() – prints something to the screen. This function accepts parameters as variables to display their values

- scanf() – receives values from the standard input and assign them to variables

Example

```
/* Calculate the area of circle */
#include <stdio.h>

int main()
{
   float r, s;

   printf(“Enter the radius of circle: ");
   scanf("%f",&r);

   s = 3.14*r*r;
   printf(“The area of circle is: s=%f", s);

   return 0;
}
```

Formatting with printf()

- Syntax

```
printf("string...",variables or numbers);
```

- The simplest use of printf is to just print out a string:

```
printf ("Hello world!");
```

- Print out a single integer number:

```
int number = 42;
printf ("Some number = %d",number);
```

Conversion character

- Conversion characters (starts with %) do not display in the screen but they are replaced by values
- Basic conversion character
  - %d: signed decimal integer
  - %u: unsigned decimal integer
  - %x: hexadecimal integer
  - %o: octal integer
  - %s: string
  - %c: single character
  - %f: fixed decimal floating point
  - %e: scientific notation floating point
- To print a character %, use %% in the format string

Print a value in different formats

- A same value can be printed in different format.
- Example

```
char ch = ‘A’;
printf ("%d\n", ch);          → print out 65
printf ("%c\n", ch);          → print out ‘A’
```

- %d is called a conversion character for integers because it tells the compiler to treat the variable to be filled into it as an integer

Print a value in different formats

```
#include <stdio.h>

int main()
{
   char c = ‘A';

   printf(“Print c in the char format: %c\n", c);
   printf(“Print c in the interger format: %d\n", c);
   printf(“Print c in the hexa format: %x", c);

   return 0;
}
```

Output:

```
Print c in the char format: A
Print c in the interger format: 65
Print c in the hexa format: 41
```

Formatting with printf

- Use special control characters such as \n, \t

- We can specify the field width to fill each data:

  % [-] [*fwidth*] [.*p*] X where:

  - [*fwidth*] the field width
  - [-] left justified.
  - [.*p*] the number of decimal places or how many characters are to be printed.

Example

```
Value     Spec.       Output
42        %6d         |    42|
42        %-6d        |42    |
'z'       %3c         |  z|
'z'       %-3c        |z  |
2.71828   %10f        |   2.71828|
2.71828   %10.2f      |      2.71|
2.71828   %-10.2f     |2.71      |
2.718     %.4f        |2.7180|
2.71828   %10e        |2.71828e+00|
"printf"  %s          |printf|
"printf"  %10s        |    printf|
```

Exercises

1. Write a program to display a menu of a restaurant, including 3 columns: meal’s code, meal’s name, price

```
            MENU
Code      Name          Price
   1      Aaa           45000.00
   2      Bbb           12500.00
```

1. Display in the screen a character and its ASCII code in the form '0': 48

scanf()

- Syntax

```
scanf ("string...",pointers);
```

- Note: Not variables which are listed after the control string, but pointers to variables.

Example:

```
int i;
char ch;
float x;
scanf ("%d%c%f", &i, &ch, &x);
// enter an integer, a character, and a real number
```

  – Notice the & characters which make the arguments pointers

Formatting with scanf

- The conversion characters for scanf are not identical to those for printf, but much more precise
    - %d : decimal integer (int)
    - %ld : long decimal integer (long)
    - %x : hexadecimal integer
    - %o : octal integer
    - %h : short integer (short)
    - %f : float type
    - %lf : long float or double
    - %c : single character
    - %s : character string

Common errors

- Find errors in the following codes:

```
float a, b, c;
scanf("%f", a);
scanf("%d", &b);
scanf("%f", &c);
```

Example

Input octal integer, output integer as decimal

```
#include <stdio.h>

int main(){
   int i ;
   scanf("%o", &i);
   printf("%d", i);
   return 0;
}
```

Input: 70
Output: 56

Exercise

Input a capital letter, output its order in alphabetical table

```
#include <stdio.h>

int main(void)
{
   char letter;

   printf(“Nhap mot ki tu thuong\n”);
   scanf(“%c”,&letter);

   printf(“Vi tri chu cai: %d\n”,letter-'a'+1);

   return 0;
}
```

Scan input data

- Values store in variables are scanned basing on input string from user. The scan process is carries out sequently and can stop when an error occurs.
- Example:

```
int i = 0;
char ch = '*';
float x = 0;
scanf ("%d%c%f ",&i,&ch,&x);
printf ("%d %c %f\n ",i,ch,x);
```

If input : 1x2.3
We have output: 1 x 2.300000

If input : 1 x 2.3
We have output: 1   0.000000

Skipping Characters in Input Stream

- Skipping blank spaces

  `scanf("%d %d %d", &day, &month, &year);`
- Skipping dashes (Enter data as dd-mm-yyyy)

  `scanf("%d-%d-%d", &day, &month, &year);`
- *Example:*

  If input is 1-1-2000, then day=1, month=1, year=2000
- As usual, if the skip string cannot be matched, scanf will abort, leaving the remaining characters in the input stream.

Return value of scanf()

- The general form of the scanf function is:

  ```
  n = scanf ("string...", pointers);
  ```

- The value n returned is the number of items matched or the end of file character EOF, or NULL if the first item did not match
- Example:

  ```
  n=scanf("%d-%d-%d", &day, &month, &year);
  ```

  - If input is 1-1-2000, then day=1, month=1, year=2000, n=3
  - If input is 1/1/2000, then day=1 and the scanf is broken, return n=1

Checking input value

```
int n;
printf("n = ");
if (scanf("%d", &n) != 1)
   printf("Can not get value for n");
```

Exercises

1. Write a program to get a character from the user and then display its ASCII code in the form '0': 48

2. Input a number and a string from the keyboard. Display them to the screen.

3. Input two time values from the keyboard and display the distance (in seconds) between them. The input time format is hh:mm:ss