Introduction to Programming

Labor 03

Revision

- Variables
- Types
- Keywords
- Operators
  - Arithmetic
  - Comparison/Relational
  - Logical
  - Bitwise
- Number systems – Conversion Rules
  - p -> 10
  - 10 -> p
  - Connection between:
    - Binary and Octal
    - Binary and Hexadecimal

Practice

- Make a new directory and enter it
  - **mkdir** labor03
  - **cd** labor03

printf()

Syntax:

- **printf** (“format string”, argument list);

Example:

```
printf (“Hello world!\n”);
                    /*displays the Hello World! text*/

printf (”a=%d\nb=%d\n”, a, b);
                    /*displays the values of a and b variables*/
```

Comments:

```
// one line comment
/* */ more line comment
```

C precedence table

```
( )   []   .  ->                                  →
*  &  +  -  !  ~  ++  --                          ←
*  /  %                                           →
+  -                                              →
>>   <<                                            →
<   >   <=   >=                                   →
==  !=                                             →
&                                                 →
^                                                 →
|                                                 →
&&                                                →
||                                                →
?:                                                →
=  +=  -=  *=  /=  %=  >>=  <<=  &=  ^=  |=       ←
,                                                 →
```

Exercise

What will be the value of the i, j and k variables after the execution of the following code.

```
int i=3, j=1, k=2;
k = i+++j;
printf("i=%d j=%d k=%d\n", i, j, k);
```

```
int i=3, j=1, k=2;
k=++i+j++;
printf("i=%d j=%d k=%d\n", i, j, k);
```

```
int i=3, j=1, k=2;
k=--i-j--;
printf("i=%d j=%d k=%d\n", i, j, k);
```

Exercise

What will be the value of the i, j and k variables after the execution of the following code.

```
int i=3, j=1, k=2;
k=-i+++j;
printf("i=%d j=%d k=%d\n", i, j, k);
```

```
int i=3, j=1, k=2;
k+=++i+--j;
printf("i=%d j=%d k=%d\n", i, j, k);
```

```
int i=3, j=1, k=2;
k+=-i+++j;
printf("i=%d j=%d k=%d\n", i, j, k);
```

Operators

Ternary operator

- condition ? value_if_true : value_if_false
- a>b ? True : False

```
int a=3, b=5, max, c;
max=a>b ? a : b;
c=a<b ? b++ : ++a;
printf("max=%d c=%d\n", max, c);
```

Exercise

What will be the value of the a, b and c variables after the execution of the following code.

Work with the newly calculated values!

```
int a, b, c;
a = b = c = 9;                printf("a=%d b=%d c=%d\n", a, b, c);
c = a++ * (b%4);              printf("a=%d b=%d c=%d\n", a, b, c);
c = a < b ? a*2 : b/3;        printf("a=%d b=%d c=%d\n", a, b, c);
b--; a++;                     printf("a=%d b=%d c=%d\n", a, b, c);
```

Operators

- **sizeof() operator**
  - determines the size of the types or the variable in byte
- **float a;**
  - sizeof(a)=4
  - sizeof(float)=4
- **double b;**
  - sizeof(b)=8
  - sizeof(double)=8

Constants in C

- Constants refer to fixed values that the program may not change during its execution.
- These fixed values are also called **literals.**
- Constants can be of any of the basic data types like:
  - an integer constant,
  - a floating constant,
  - a character constant, or
  - a string literal.
- There are enumeration constants as well.
- Constants are treated just like regular variables except that their values **cannot be modified** after their definition.
- There are two simple ways in C to define constants:
  - Using **#define** preprocessor.
  - Using **const** keyword.

Integer constants (literals)

- An integer constant is a numeric constant (associated with number) without any fractional or exponential part :
  - Decimal constants: 0, –9, 22 etc.
  - Octal constants: 021, 077, 033 etc.
  - Hexadecimal constants: 0x7f, 0x2a, 0x521 etc.
  - (Octal constant starts with a 0 and hexadecimal constant starts with a 0x.)

- For example:
  - 85 /* decimal */
  - 0213 /* octal */
  - 0x4b /* hexadecimal */
  - 30 /* int */
  - 30u /* unsigned int */
  - 30l /* long */
  - 30ul /* unsigned long */

Floating-point constants

- A floating-point constant is a numeric constant that has either a fractional form or an exponent form.

For example:

- -2.0
- 0.0000234
- -0.22E-5

- Note: **E-5 = 10^{-5}**

Character constants

- A character constant is a constant which uses single quotation (') around characters.

For example:

- 'a', 'l', 'm', 'F'

String constants

- String constants are the constants which are enclosed in a pair of double-quote marks (").

For example:

- "good" //string constant
- "" //null string constant
- " " //string constant of six white space
- "x" //string constant having single character
- "Earth is round\n" //prints string with newline

Enumeration constants

- Keyword **enum** is used to define enumeration types.

For example:

- **enum** color {yellow, green, black, white};

- color is a variable and yellow, green, black and white are the enumeration constants having value 0, 1, 2 and 3 respectively.

Constant in C – The #define Preprocessor

- Given below is the form to use **#define** preprocessor to define a constant:

#define identifier value

- For example:
  - #define N 10
  - #define PI 3.141593
  - #define TRUE 1
  - #define FALSE 0

Constant in C – The const Keyword

- You can use **const** prefix to declare constants with a specific type as follows:
  - **const** type variable = value;
- For example:
  - **const int** LENGTH = 10;
  - **const float** WIDTH = 5.25;
  - **const char** NEWLINE = '\n';

Exercise

What is the result of this code?

```
int a, b;
a=057;
b=024;
a= a & b;

printf("a=%d\n", a);
printf("a=%o\n", a);
printf("a=%X\n", a);
```

Octal

Exercise

What is the result of this code?

```
int a, b;
a=0x2B;
b=0x31;
a= a | b;

printf("a=%d\n", a);
printf("a=%o\n", a);
printf("a=%X\n", a);
```

Hexadecimal

Exercise

What is the result of this code?

```
int a, b;
a=0x6F;
b=0x57;
b |= (1<< 3);

printf("b=%d\n", b);
printf("b=%o\n", b);
printf("b=%X\n", b);
```

Commonly used escape sequences

\n	newline
\t	tab
\v	vertical tab
\f	new page
\b	backspace
\r	carriage return
\o	null character
\?	to print question mark
\\	to print slash
\'	to print single quote
\"	to print double quote

scanf()

- **scanf** (“formatted _specifier”, &variable_ name);
- **&** (address operator)

Example

```
int a, b, i;
float j;

scanf(”%d”, &a);

scanf(”%d %d”, &a, &b);

scanf(“%d %f”, &i, &j);
```

Exercise

- Write a program to input two integer numbers and print their sum, product, difference and quotient?

Solution

```
#include <stdio.h>
int main()
{
    int a, b;

    printf("a=");
    scanf("%d", &a);

    printf("b=");
    scanf("%d", &b);

    printf("sum: %d\n", a+b);

    printf("product: %d\n", a*b);

    printf("difference: %d\n", a-b);

    printf("quotient: %d\n", a/b);
    //printf("quotient: %.2f\n", a/(float)b);


    return 0;
}
```

Exercise

Write a program to input two integer numbers into two variables, change the content of the variables with each other and print the variables in inverse order.

Solution A – using auxiliary variable

Solution B – using arithmetic operators (+, –)

Solution C – using bit operators (^)

Solution A

```
int a, b, tmp;

printf("a=");    scanf("%d",&a);
printf("b=");    scanf("%d",&b);

     tmp=a;
     a=b;
     b=tmp;

printf("a=%d b=%d\n", a, b);
```

Solution B

```
int a, b;

printf("a=");    scanf("%d",&a);
printf("b=");    scanf("%d",&b);

     a=a+b;  //a+=b;
     b=a-b;
     a=a-b;  //a-=b;

printf("a=%d b=%d\n", a, b);
```

Solution C

```
int a, b;

printf("a=");    scanf("%d",&a);
printf("b=");    scanf("%d",&b);

        a=a^b;  //a^=b;
        b=a^b;  //b^=a;
        a=a^b;  //a^=b;

printf("a=%d b=%d\n", a, b);
```

Exercise – Homework!

What will be the value of the a, b and c variables after the execution of the following code.

```
int a = 15, b = 15, c = 15;
c = (a%2) + (a=!b);  printf("a=%d b=%d c=%d\n", a, b, c);

int a = 2, b = 5, c = 15;
c = a < b ? ++a : b++; printf("a=%d b=%d c=%d\n", a, b, c);

int a = 2, b = 15, c = 1;
b=4/3*c*c; a=b!=a; printf("a=%d b=%d c=%d\n", a, b, c);
```

Exercise – Homework!

What will be the value of the a, b and c variables after the execution of the following code.

```
int a = 15, b = 15, c = 15;
c = (a%2) + (a!=b);  printf("a=%d b=%d c=%d\n", a, b, c);

int a = 2, b = 5, c = 15;
c = a > b ? ++a : b++; printf("a=%d b=%d c=%d\n", a, b, c);

int a = 2, b = 15, c = 1;
b=4/3*c*c; a=b=!a; printf("a=%d b=%d c=%d\n", a, b, c);
```