King Fahd University of Petroleum and Minerals
Information & Computer Science Department
ICS 431 Operating System Lab
Lab # 2
Introduction to C language

Objectives:

- To know what is importance of C-Language
- To know syntax of Unix C-Program
- To know how to create, compile, and execute a C program under Unix System.

What is importance of C-Language?

C language is known as middle level language, which is having features of low level, as well as higher-level languages. That's why C language is very much suitable for the systems programming. C language is still most suitable language for systems programming.

Sample Examples of C-Programs:

From the following sample C-Programs you can learn syntax of C-Program, therefore, please run following sample programs in your computer using **pico** editor and understand syntax of each sample program:

Sample#1: (Use of if-statement, relational & logical operators)

```
#include<stdio.h>
main()
{
        int a,b,c,max;
        printf("Please Input three Numbers: "); //to display message on screen
        scanf("%d %d %d",&a,&b,&c); //to read input from keyboard

        if(a>b && a>c) //use of if-else-if statement and logical operators
        max=a;
        else if(b>a && b>c)
        max=b;
        else
        max=c;

        printf("\nThe maximum of  %d, %d, %d is = %d\n",a,b,c,max); //to display output
}
```

How to type and run this program in Unix Environment:

```
vlsi> gcc s1.c -o s1
vlsi> s1
Please Input three Numbers: 1 5 8

The maximum of  1, 5, 8 is = 8
vlsi>
```

Note:

The Syntax of different forms of if-statements, switch -statement, for-loop, while-loop, do-while loop is exactly same as in java language. Use of relational & logical operators in C language is also exactly same as in java language.

Sample#2: (Use of Math Library Functions)

```
#include<stdio.h>  // input output standard functions are defined in this
#include<math.h>  // all standard math functions are defined
#include<stdlib.h>  // here used only for rand() function
main()
{
int ab,random;   // variable declarations
double cel,cosine,floatabs,flo,logbase_e,logbase_10,power,expo,sq_root,ncel,nflo;

        ab=abs(-8);  // gives absolute value of integer argument
        cel=ceil(45.0001);   // gives ceil value of double argument
        ncel=ceil(-45.0001);   // gives ceil value of double argument
        cosine=cos(30*3.14159/180);  //  this first angle must be converted in radian
        expo=exp(1.0);     // gives  e  to the power argument value
        floatabs=fabs(-8.432); // gives absolute value of float argument
        flo=floor(45.99356);  // it truncates decimal digits
        nflo=floor(-45.99356);  // it truncates decimal digits
        logbase_e=log(2.71828); // calculates natural log of argument (i.e base e)
        logbase_10=log10(100); // calculates log of argument at base 10
        power=pow(0.16,0.5); // calculates power i.e pow(base, power) format
        random=rand(); // generates random number
        sq_root=sqrt(2.25); // calculates squirroot of argument

        printf("The absolute value of -8 is :  %d\n", ab);
        printf("The ceil value of 45.0001 is : %lf\n", cel);
        printf("The ceil value of -45.0001 is : %lf\n", ncel);

        printf("The cos value of degree 30  is : %lf\n", cosine);
        printf("The exponential  value of 1 is : %lf\n", expo);
        printf("The absolute  value of double type argument -8.432 is : %lf\n", floatabs);
        printf("The floor  value of 45.99356  is : %lf\n",  flo);
        printf("The floor  value of -45.99356  is : %lf\n",  nflo);
        printf("The natural log  value of 2.71829 is : %lf\n",  logbase_e);
        printf("The log base 10 for  value 100 is : %lf\n",   logbase_10);
        printf("The  value of pow(0.16,0.5)  is : %lf\n",  power);
        printf("The random number is  : %d\n", random);
        printf("The  value of sqrt(2.25)  is : %lf\n",  sq_root);

} // end of main
```

Sample output:

```
vlsi> gcc s2.c -o s2 -lm
vlsi> s2
```

```
The absolute value of -8 is :  8
The ceil value of 45.0001 is : 46.000000
The ceil value of -45.0001 is : -45.000000
The cos value of degree 30  is : 0.866026
The exponential  value of 1 is : 2.718282
The absolute  value of argument -8.432 is : 8.43200
The floor  value of 45.99356  is : 45.000000
The floor  value of -45.99356  is : -46.000000
The natural log  value of 2.71829 is : 0.999999
The log base 10 for  value 100 is : 2.000000
The  value of pow(0.16,0.5)  is : 0.400000
The random number is  : 16838
The  value of sqrt(2.25)  is : 1.500000
vlsi>
```

Note:

To run any Unix C-Program in which we use Standard Math library Function we must link math library by using **–lm** as used above.

Sample#3: (How to read /write using files in Unix C Program)

```
#include <stdio.h>
#include<stdlib.h>
#define PI 3.14159
main(void) {
        double radius, area, circum;
        FILE  *inp, *outp;  //file pointers
       inp = fopen("circle.dat", "r"); //open file in read mode
       if(inp==NULL) {
              printf("Not able to open input file");
              exit(1); // terminates the program
       }
       outp = fopen("circle.out", "w"); // open file in write mode
       fscanf(inp, "%lf", &radius); // read data from file
       fprintf(outp, "The radius is %.2f\n", radius); // write output in the file
       area = PI * radius * radius;
       circum = 2 * PI * radius;
       fprintf(outp, "The area is %.2f\n", area);
       fprintf(outp, "The circumference is %.2f\n", circum);
       fclose(inp); // closes input file
       fclose(outp); // closes output file
       return (0);
} // end of main
```

Sample output:

```
vlsi> gcc s3.c -o s3
vlsi> s3
```

vlsi> cat circle.out
The radius is 1.00
The area is 3.14
The circumference is 6.28
vlsi>

Sample#4: (How to write & use user-defined function (method in java))

```
#include<stdio.h>
float mul(float x, float y) // user defined function mul
{
        float p;
        p=x*y;
        return(p);
} // end of mul function


float division(float number1, float number2) // user defined function division
{
        float div;
        div=number1/number2;
        return div;
} // end of division function


main( ) // calling function
{
        float n1, n2, product, d;
        printf("Please input value of n1 and n2 : ", n1, n2);
        scanf("%f %f", &n1, &n2);

        product = mul(n1, n2); // function call
        d = division(n1, n2);  // function call

        printf("The product of %0.2f and %0.2f is = %0.2f\n", n1, n2, product);
        printf("The division of %0.2f and %0.2f is = %0.2f\n", n1, n2, d);

} // end of main
```

Sample Output:

vlsi> gcc s4.c -o s4
vlsi> s4
Please input value of n1 and n2 : 4 2
The product of 4.00 and 2.00 is = 8.00
The division of 4.00 and 2.00 is = 2.00
vlsi>

Sample#5: (How to write recursive function)

```
#include<stdio.h>
long double factorial (int n) // user defined recursive function
{
        if (n == 0)
           return 1;
        else
           return( n * factorial (n-1));
}


main( ) // calling function
{
        int x;
        long double fact;
        printf("Please input value of x : ");
        scanf("%d",&x);
        fact = factorial(x); // function call

        printf("The Factorial of %d is %Lf", x, fact);

} // end of main
```

Sample Output:

```
vlsi> gcc s5.c -o s5
vlsi> s5
Please input value of x : 5
The Factorial of 5 is 120.000000
vlsi>
```

The C Preprocessor:

The preprocessor provides facilities for including source code from other files (useful in separate compilation and when using library routine) and for doing conditional compilation (which is used frequently when writing portable code). It also provides a powerful macro facility, which can be very useful to declare symbolic constants.

Any preprocessor command starts with pound sign (#) which forms the beginning of the preprocessor command. The **carriage return** forms the end of any preprocessor command. Now, we will consider only two of the C preprocessors' capabilities; text inclusion, and simple text substitution.

Text inclusion:

Look at this program:

```
void main (void)
{
    printf("Hello World!\n");
}
```

Notice that the printf function has not been defined anywhere. The printf is part of a set function used for standard input and output. These functions are defined in a separate file known as the standard library. In order to use these functions correctly, a source code file requires information about their parameters and return type. This information is provided in a header file called stdio.h. All the information in the stdio.h can be brought into a source code file by use of a preprocessor include operation.

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

The file name is stdio.h ('h' for a 'h'eader file) and the fact it is surrounded by angle brackets ('<...>') indicates to the preprocessor that it is to search in the system "include file" directories to find the header file. It is possible to create your own header files and include them.

You can surround the header file by double quotes ("...") ,instead of angle brackets. In this case, the preprocessor will search in the directory where it found the source file, which is including the header file. If the header file is not found there, it will then search the system include file directories as well.

Simple text substitution:

The preprocessor #define is used to do text substitution. It is important to always remember that this is a very simple-minded facility, which simply replaces one text string where it finds an occurrence of another. It is thus easy to get into trouble if you are not careful. The define command consists of the define keyword, a define symbol, and a replacement text.

Example:

```
#define  message   "Hello world!\n"
int main()
{
   printf(message);
}
```

When programming in C, mostly we make symbolic constants by using #define command. For example

```
#define   FALSE    0
#define   TRUE     1
```

Notice that there are no semicolons at the end of the preprocessor statements. This is because they are not statements. If you want to learn more about the C preprocessor see 'man cpp'.

Exercises

Note:
Lab Problems will be given during the lab based on material covered in this lab manual.