CSE321 - Operating Systems Lab

Lab 02: Introducing C Programming (01)

Summary

- How to write a C program
- Data types in C
- Pointer in C
- Array in C
- String in C

How to write a C program:

A First C Program

Consider the following C program

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

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

Assume that it is stored in a file called `hello.c` in the current working directory. Then:

```
$ gcc -o hello hello.c
```
(Note: Compiles without warning or error)
```
$ ./hello
Hello World!
```

Data types in C

Type	Description
Char	Typically a single octet(one byte). This is an integer type.
Int	The most natural size of integer for the machine.
Float	A single-precision floating point value.
Double	A double-precision floating point value.
Void	Represents the absence of type.

Declare variables:

```
int    i, j, k;
char   c, ch;
float  f, salary;
double d;
```

```
extern int d = 3, f = 5;    // declaration of d and f.
int d = 3, f = 5;           // definition and initializing d and f.
byte z = 22;                // definition and initializes z.
char x = 'x';               // the variable x has the value 'x'.
```

```
#include <stdio.h>
int main ()
{
// Variable definition:
int a, b;
int c;
float f;
// actual initialization
a =10;
b =20;
c = a + b;
printf("value of c : %d \n", c);
f = 70.0/3.0;
printf("value of f : %f \n", f);
return 0;
}
```

Taking Inputs

scanf(const char *format, ...) function reads input from the standard input stream stdin and scans that input according to format provided.

```c
#include <stdio.h>
int main( )
{
   char str[100];
   int i;

   printf( "Enter a value :");
   scanf("%s %d", str, &i);

   printf( "\nYou entered: %s, %d ", str, i);

   return 0;
}
```

```
Enter a value : seven 7
You entered: seven 7
```

Character input:

```c
#include <stdio.h>
int main( )
{
   int c;

   printf( "Enter a value :");
   c = getchar( );

   printf( "\nYou entered: ");
   putchar( c );

   return 0;
}
```

When the above code is compiled and executed, it waits for you to input some text when you enter a text and press enter then program proceeds and reads only a single character and displays it as follows:

```
$./a.out

Enter a value : this is test
You entered: t
```

Arrays in C

```c
#include <stdio.h>
int main ()
{
 int n[ 10 ]; /* n is an array of 10 integers */
 int i,j;
 /* initialize elements of array n to 0 */
 for ( i = 0; i < 10; i++ )
   {
    n[ i ] = i + 100; /* set element at location i to i + 100 */
   }
 /* output each array element's value */
 for (j = 0; j < 10; j++ )
   {
    printf("Element[%d] = %d\n", j, n[j] );
   }
 return 0;
}
```

output -
Element[0] = 100
Element[1] = 101
Element[9] = 109

Multidimensional Arrays:

```c
#include <stdio.h>
int main ()
{
/* an array with 5 rows and 2 columns*/
 int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
 int i, j;
/* output each array element's value */
 for ( i = 0; i < 5; i++ )
  {
    for ( j = 0; j < 2; j++ )
     {
        printf("a[%d][%d] = %d\n", i,j, a[i][j] );
     }
  }
 return 0;
}
```

a[0] a[0] = 0
a[0] a[1] = 1
a[1] a[0] = 1

Pointers in C

A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location. Following are the valid pointer declaration:

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

An example:

```
#include <stdio.h>

int main ()
{
   int  var = 20;   /* actual variable declaration */
   int  *ip;        /* pointer variable declaration */

   ip = &var;  /* store address of var in pointer variable*/

   printf("Address of var variable: %x\n", &var  );

   /* address stored in pointer variable */
   printf("Address stored in ip variable: %x\n", ip );

   /* access the value using the pointer */
   printf("Value of *ip variable: %d\n", *ip );

   return 0;
}
```

Output will be -

```
Address of var variable: bffd8b3c

Address stored in ip variable: bffd8b3c

Value of *ip variable: 20
```

Incrementing a Pointer

We prefer using a pointer in our program instead of an array because the variable pointer can be incremented, unlike the array name which cannot be incremented because it is a constant pointer. An example is -

```
#include <stdio.h>

const int MAX = 3;

int main ()
{
   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have array address in pointer */
   ptr = var;
   for ( i = 0; i < MAX; i++)
   {

      printf("Address of var[%d] = %x\n", i, ptr );
      printf("Value of var[%d] = %d\n", i, *ptr );

      /* move to the next location */
      ptr++;
   }
   return 0;
}
```

The output will be -

```
Address of var[0] = bf882b30

Value of var[0] = 10

Address of var[1] = bf882b34

Value of var[1] = 100

Address of var[2] = bf882b38

Value of var[2] = 200
```

Strings in C:

Strings are an array of characters ending with NULL in C.

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

Example program:

```
#include <stdio.h>

int main ()
{
   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

   printf("Greeting message: %s\n", greeting );

   return 0;
}
```

Output will be-

```
Greeting message: Hello
```

Functionalities provided by C -

S.N.	Function & Purpose
1	**strcpy(s1, s2);** Copies string s2 into string s1.
2	**strcat(s1, s2);** Concatenates string s2 onto the end of string s1.
3	**strlen(s1);** Returns the length of string s1.
4	**strcmp(s1, s2);** Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	**strchr(s1, ch);** Returns a pointer to the first occurrence of character ch in string s1.
6	**strstr(s1, s2);** Returns a pointer to the first occurrence of string s2 in string s1.

Struct in C:

The struct statement defines a new data type, with more than one member for your program. The format of the struct statement is -

```
struct [structure tag]
{
   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

An example program with struct -

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

struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

int main( )
{
   struct Books Book1;        /* Declare Book1 of type Book */
   struct Books Book2;        /* Declare Book2 of type Book */

   /* book 1 specification */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming Tutorial");
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Zara Ali");
   strcpy( Book2.subject, "Telecom Billing Tutorial");
   Book2.book_id = 6495700;

   /* print Book1 info */
   printf( "Book 1 title : %s\n", Book1.title);
   printf( "Book 1 author : %s\n", Book1.author);
   printf( "Book 1 subject : %s\n", Book1.subject);
   printf( "Book 1 book_id : %d\n", Book1.book_id);

   /* print Book2 info */
   printf( "Book 2 title : %s\n", Book2.title);
   printf( "Book 2 author : %s\n", Book2.author);
   printf( "Book 2 subject : %s\n", Book2.subject);
   printf( "Book 2 book_id : %d\n", Book2.book_id);

   return 0;
}
```

The output will be -

```
Book 1 title : C Programming
Book 1 author : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407
Book 2 title : Telecom Billing
Book 2 author : Zara Ali
Book 2 subject : Telecom Billing Tutorial
Book 2 book_id : 6495700
```

Lab Tasks:

1. Write a C program to print your name, date of birth. and mobile number.
2. Write a C program to compute the perimeter and area of a rectangle with a height of 7 inches. and width of 5 inches.
3. Write a C program to compute the perimeter and area of a circle with a given radius.
4. Write a program in C to store elements in an array and print it.
5. Write a program in C to read n number of values in an array and display it in reverse order
6. Write a C program to convert a given integer (in seconds) to hours, minutes and seconds.
7. Write a C Program to Swap Values of Two Variables.
8. Write a C Program to Print ASCII Value of a given user input.
9. Write a C program to concatenate two strings given by the user.
10. Write a C program to store students information in a struct. Take input from users for a student and print it.