Data Types in C

Each variable in C has an associated data type. Each data type requires different amounts of memory and has some specific operations which can be performed over it

Below is list of ranges along with the memory requirement and format specifiers on ***32 bit gcc compiler***

DATA TYPE	MEMORY (BYTES)	RANGE	FORMAT SPECIFIER
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4		%f
double	8		%lf
long double	12		%Lf

Variable Declarations

Every variable occurring in a program must be declared before it is used.

The syntax for declaring a variable in C is :

var_type *list variables*;

ex. **int** i,j,k; **float** x,y,z; **char** letter;

Constants

C allows you to declare ***constants***. Value defined as a constant **can not be changed** afterwards. The **const** keyword is to declare a constant, as shown below:

const int a =2;

Another way to define constants in a program is by using **#define** , as shown below.

#define a 2

Global Variables

Global variables are defined above **main()** in the following way:

```
short number,sum;
int bignumber,bigsum;
char letter;
main() {
        }
```

It is also possible to **pre-initialise** global variables using the = operator for assignment.

Example

```
float sum=0.0;
int bigsum=0;
char letter=`A';
main () { }
```

Functions

Modules in C are called functions. There are two types of functions:

1.) *Programmer-defined* functions – one that programmer writes
2.) *Pre-packaged* functions available in the C standard library (**printf, scanf** etc.)

Functions are invoked by a *function call.* The function call specifies the function name and provides information (as arguments) that the called function needs in order to perform its designated task.

All variables declared in function definitions are *local variables* and they are **known only in the function in which they are defined.**

The format of a function definition is:

Type of the result returned to caller (void – no value returned

return-value-type *function-name* (**parameter-list**)
{
declarations

statements
}

Function Body

Comma-separated list; type must be listed for each parameter except for ***int***

Examples 1

Write a function that takes two parameters as input and returns the summation of the two input numbers

```
#include <stdio.h>
int addition(int num1, int num2)
{
     int sum;
          sum = num1+num2;

     /* Function return type is integer so we are returning
      * an integer value, the sum of the passed numbers.
      */
     return sum;
}

int main()
{
     int var1, var2;
     printf("Enter number 1: ");
     scanf("%d",&var1);
     printf("Enter number 2: ");
     scanf("%d",&var2);

     /* Calling the function here, the function return type
      * is integer so we need an integer variable to hold the
      * returned value of this function.
      */
     int res = addition(var1, var2);
     printf ("Output: %d", res);

     return 0;
}
```

Examples 2

Creating a void user defined function that doesn't return anything

```
#include <stdio.h>
/* function return type is void and it doesn't have parameters*/
void introduction()
{
    printf("Hi\n");
    printf("My name is John\n");
    printf("How are you?");
    /* There is no return statement inside this function, since
its
     * return type is void
     */
}

int main()
{
     /*calling function*/
     introduction();
     return 0;
}
```

Examples 3

Write a function that takes two parameters as input and returns the max of the two input numbers

```
#include <stdio.h>
/* function declaration */
int max(int num1, int num2);
int main () {
   /* local variable definition */
   int a = 100;
   int b = 200;
   int ret;
   /* calling a function to get max value */
   ret = max(a, b);
   printf( "Max value is : %d\n", ret );
   return 0;
}
/* function returning the max between two numbers */
int max(int num1, int num2) {

   /* local variable declaration */
   int result;
   if (num1 > num2)
      result = num1;
   else
      result = num2;
   return result;
}
```

Example 4

Write a function that takes one parameter 'x' as input and returns the square of the input value.

```
#include<stdio.h>
// function declaration
float square ( float x );
// main function, program starts from here

int main( )
{
    float m, n ;
    printf ( "\nEnter some number for finding square \n");
    scanf ( "%f", &m ) ;
    // function call
    n = square ( m ) ;
    printf ( "\nSquare of the given number %f is %f",m,n );
}

float square ( float x )   // function definition
{
    float p ;
    p = x * x ;
    return ( p ) ;
}
```

Example 5

Write a function that takes two parameters as input and returns the swap of the two input values

```
#include<stdio.h>
// function prototype, also called function declaration
void swap(int a, int b);
int main()
{
    int m = 22, n = 44;
    // calling swap function by value
    printf("values before swap  m = %d and n = %d", m, n);
    swap(m, n);
}
void swap(int a, int b)
{
    int tmp;
    tmp = a;
    a = b;
    b = tmp;
    printf("\nvalues after swap m = %d and n = %d", a, b);
}
```

Example 6

Write a function that takes one parameter 'x' as input and returns the absolute value of the input

```
#include <stdio.h>

/*this is the function to compute the absolute value of a whole
number.*/

int abs(int x)

{

        if (x>=0) return x;

        else return -x;

}

/*this is the program that uses twice the function defined
above.*/

int main()

{

        int x, y;

  printf("Type the coordinates of a point in 2-plane, say P =
(x,y). First x=");

        scanf("%d", &x);

        printf("Second y=");

        scanf("%d", &y);

 printf("The distance of the P point to the x-axis is %d. \n Its
distance to the y-axis is %d. \n", abs(x), abs(y));

        return 0;

}
```

Example 7

The following example illustrate the usage of a function as a procedure. Consider you are making a program that tells students if they are approved in the discipline or not.

```
#include<stdio.h>
void check(int x)
{
if (x<60)
printf("Sorry! You will need to try this course again.\n");
else
printf("Enjoy your vacations! You are approved.\n");
}
/*here the program actually starts, and use the check function three times.*/
int main()
{
 int a, b, c;

printf("Type your grade in Mathematics (whole number). \n");
scanf("%d", &a);
check(a);
printf("Type your grade in Science (whole number). \n");
scanf("%d", &b);
check(b);
printf("Type your grade in Programming (whole number). \n");
scanf("%d", &c);
check(c);
return 0;
}
```

Recursive Functions

As it was said before modules in C are called functions. One of the types of functions is a recursive function. A recursive function is a function that calls itself for further simplified processing. As the function calls itself the task is divided into more manageable subtasks. The recursive function calls continuously till a known base-case is found. A typical recursive function has a recursive formula and a base case.

Example: Consider the factorial problem:

n!=n*(n-1)! **(Recursive formula)**
1!=1*(0!) **(Base-case)**

The recursive function that can be written for calculating a recursive function is as follow:

```
int factorial (int a)
{       if ( (a==0) || (a==1) )
                return 1;
        else
                return ( a * factorial(a-1) );
```

Example 8

find factorial of a non-negative integer (entered by the user) using recursion.

```
#include <stdio.h>
long int multiplyNumbers(int n);
int main()
{
   int n;
   printf("Enter a positive integer: ");
   scanf("%d", &n);
   printf("Factorial of %d = %ld", n, multiplyNumbers(n));
   return 0;
}
long int multiplyNumbers(int n)
{
   if (n >= 1)
      return n*multiplyNumbers(n-1);
   else
      return 1;
}
```

Homework

1.a (Due after one week)

Write a program that inputs 5 real numbers from the keyboard, and prints the **sum**, **mean**, **median**, **product**, **smallest**, and **largest** of these numbers.

1.b (Due after one week)

Write a program that inputs three real numbers. Then write a **function** called **Total** to find the output of the following operation. If any of the input numbers are negative the programme should print an error message and exit, otherwise the program calculates and prints the output according to the formula.

$$Q = \left\{ \frac{1}{2}a^2 + 3b^3 - \frac{5c}{4} \right\}$$

a , b and c are entered by the user.

Notes

Please Send your Homework in the following Email

eeng212laboratory@gmail.com

Subject of email Should include student Number + Homework Number for Example: "St. 15000012 Homework #1