Laboratory 2: Cover Sheet

Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Roll.No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Semester \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Constant, Variable, Keywords, Data Types and input statement in C Language

|  |  |  |
| --- | --- | --- |
| **Activities** | **Remarks** | **Signature** |
|  |  |  |

**LAB2**

# Constant, Variable, Keywords, Data Types and input statement in C Language

#### Objectives:

Familiarization with Constants, Variables, and Keywords. To understand the concept of using data types in C language.

To understand the concept of declaration of data type (declaration statement) To manipulate various data types available in C language.

To understand data type conversion. Utilization of input and output function To be familiar with format specifier

1. **Fundamentals of Data Storage**

Data is stored in a computer's memory. The memory system comprises of uniquely numbered cells called memory addresses. We need to know the address where something is stored in order to retrieve it and work on it. A programming language frees us from keeping track of these memory addresses by substituting names for them. These names are called variables. Variables are descriptive names for the memory addresses.

## The C Character Set

A character denotes any **alphabet**, digit or **special symbol** used to represent information. Below figure shows the valid alphabets, numbers and special symbols allowed in C.

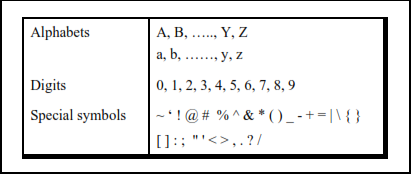


Figure 1 – The C character set

## Constants, Variables and Keywords

* 1. **Constants:** are values that do not change its value i.e. “3.1415”. A constant’s value does not change at the time of execution of a program.
  2. **Variables:** are entities that may change. Both variables and constants, is stored as a sequence of binary digits, e.g. strings of zeros and ones. These binary digits are referred to as

"bits". The *"type"* of a particular variable or constant determines how many bits are used for that particular data item, and how the bits are to be interpreted.

* 1. **Keywords:** Keywords are the words whose meaning has already been explained to the C compiler. The keywords **cannot** be used as variable names because if we do so we are trying to assign a new meaning to the keyword, which is not allowed by the computer.

Following are some examples of some keywords

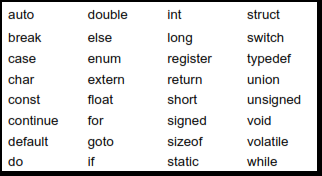


Figure 2 – The C Keywords

Before we use a variable in C we must declare it. We must identify what kind of information will be stored in it. This is called defining a variable. Variables must be declared at the start of any block of code, but most are found at the start of each function. A variable must be defined to be one of the legal C data types. When a variable is defined it is not automatically initialized, it is the responsibility of the programmer to initialize this to a start value.

## Using DEFINE Keyword

The preprocessor #define is more flexible method to define constants in a program.

|  |  |
| --- | --- |
| #define TRUE | 1 |
| #define FALSE | 0 |
| #define NAME\_SIZE | 20 |
| **Example - 1:** |  |

#include <stdio.h> #define LENGTH 10

#define WIDTH 5 #define NEWLINE '\n'

int main()

{

clrscr();

printf("value of length : %d", LENGTH); printf("%c", NEWLINE);

printf("value of width: %d",WIDTH); printf("%c", NEWLINE);

getch();

}

#### Task: Write a program to print your name and roll number using #DEFINE keyword

1. **Data types**

All programs are composed of two items: Data and Operations on that Data. As Kernighan and Ritchie say,

*“The type of an object determines the set of values it can have and what operations can be performed on it.”*'

Basic C recognizes three categories of data types: Integer, Floating Point and Character. Modern C adds a few special types to this list.

|  |  |
| --- | --- |
| **Datatypes** | **Meanings** |
| **Integer** | A whole numbers that is numbers not having any fractional components |
| **Float** | An integer with floating point |
| **Character** | A single character |

* 1. **Integers**

Table 1 – Data types in C Language

#### int

* + The most basic and commonly used integral type is "int".
  + The int data type is always the "best" size for the particular computer it is running on, typically 32 bits.

#### long int

* + A long int typically uses twice as many bits as a regular int, allowing it to hold much larger numbers.
  + long int may also be specified as just long.

#### short int

* + A short int may use fewer bits than a regular int, thereby saving storage space.
  + short int may also be specified as just short.

#### Unsigned ints

* + Unless otherwise specified, all of the aforementioned int types are signed numbers (contain only positive values)

### Floating Points

Floating point types include all types in which a number may have a fractional component. Fortunately there are only three that we need to worry about-float, double, and long double.

#### float

* + The most basic type of floating point number is the float type.
  + Note that a float only has 24 bits ( 4 b ytes) available for storing the digits of the number

#### double

* + The double precision data type uses twice as many bits as a float, yielding approximately twice the number of digits of precision.
  + The double data type is the preferred floating point type for most scientific and engineering calculations.

#### Long double

* + The long double type is guaranteed to have more bits than a double, but the exact number may vary from one hardware platform to another. The most typical implementations are either 80 or 128 bits (16 bytes).

### Characters

#### char

* + Normally chars are interpreted as characters
  + Technically the char data type is an integral type, always having exactly 8 bits (1 byte).

## Data Type Declaration

Any variable used in the program must be declared before using it. For example,

int p, n ; float r, si ; char ch

Data type Keyword

Variable Name

Figure 3 – Data type declaration in C Language

### Syntax of Data Type Declaration

data\_type variable\_name ;

|  |  |  |  |
| --- | --- | --- | --- |
| **Data types** Bytes Format **Range** | | | |
| short int | 2 | %d | -32,768 -> +32,767 |
| unsigned int | 2 | %u | 0 -> 65535 |
| int | 2 | %d | -32,768 -> +32,767 |
| long int | 4 | %ld | -2,147,483,648 -> +2,147,483,647 |
| char | 1 | %c | 0 -> +255 |
| float | 4 | %f | -3.4e38 -> +3.4e38 |
| double | 8 | %lf | -1.7e308 -> +1.7e308 |
| long double | 12 | %lf | -1.7e4932 -> +1.7e4932 |

Table 2 – Data types in C

### Example - 2:

|  |
| --- |
| **int Count;**  **Count = 5;** |
| **float Miles;**  **Miles = 5.6;** |
| **char Letter;**  **Letter = 'x'** |
| **double Atoms;**  **Atoms = 2500000;** |

Figure 3 – Declaration Example

## Integer and Float Conversions

Rules that are used for the implicit conversion of floating point and integer values in C:

1. An arithmetic operation between an integer and integer always yields an integer result.
2. An operation between a real and real always yields a real result.
3. An operation between an integer and real always yields a real result. In this operation the integer is first promoted to a real and then the operation is performed. Hence the result is real.

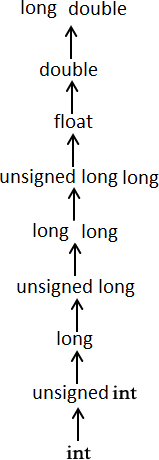
The usual arithmetic conversions are implicitly performed to cast their values in a common type. Compiler first performs *integer promotion*; if operands still have different types then they are converted to the type that appears highest in the following hierarchy:

Figure 4 – Conversion Hierarchy

### Example - 3:

Familiarization with integer, float and char with format specifier: #include <stdio.h>

int main(void)

{

inti\_value = 16777217;

floatf\_value = 16777216.0; charch = ‘Y’;

printf("The integer is: %d\n", i\_value); printf("The float is: %f\n", f\_value); printf("The Character is: %c\n",ch);

}

#### Output:

The integer is: 16777217

The float is: 16777216.000000 The Character is: Y

## Receiving Input

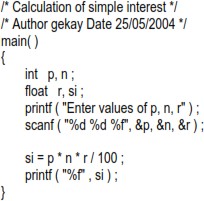
In the above we assumed the values of **p**, **n** and **r** to be 1000, 3 and 8.5. Every time we run the program we would get the same value for simple interest.

To make the program general, the program itself should ask the user to supply the values of **p**, **n**

and **r** through the keyboard during execution.

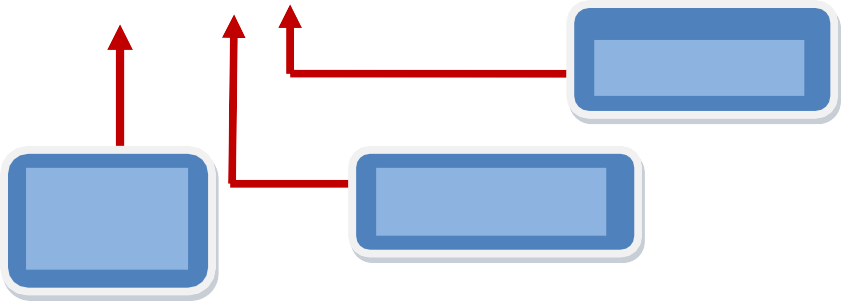
This can be achieved using a function called **scanf( )**. This function is a counter-part of the **printf()** function. **printf()** outputs the values to the screen whereas **scanf()** receives them from the keyboard. This is illustrated in the program shown below.

### Example - 4:



Outputs the message ‘Enter values of p, n, r’ the screen

Ampersand (**&**) before the variables in the **scanf()** function is a must. **&** is an ‘Address of’ operator



Data type

Address of num

scanf() function

Format specifier

void main(void)

{

int num;

scanf( "%d”, &num");

}

User defined variable name

Figure 5 – Scanf() explained

### Example - 5:

main( )

{

}

intnum ; clrscr();

printf ( "Enter a number" ) ; scanf ( "%d", &num ) ;

printf ( "Now I am letting you on a secret..." ) ;

printf ( "You have just entered the number %d", num ) ;

Write the above program on Turbo C and write output below

## Format Specifiers:

The "Format specifier" is the sequence passed as the formatting string argument;"Characters matched" gives the format of the sequence sought or printed.

|  |  |  |
| --- | --- | --- |
| **Format Specifieres** |  | **How it use** |
| %d | For Integer number (Decimal) | scanf(“%d”, int) |
| %f | For Floating point numbers | scanf(“%f”, float) |
| %c | For Character | scanf(“%c”, char) |
| %s | For String | scanf(“%s”, char[]) |
| %u | For unsigned number (Positive) | scanf(“%u”, int) |
| %x | For Hexa-decimal number | scanf(“%x”, int) |
| %e | For scientific notation | scanf(“%e”, int) |

Table 3 – Format specifiers for data types

### Example - 6:

void main (void)

{

printf(“My house is %d miles away from %s”,2, “UIT ”); printf(“The letter %c is pronounced as %s”, ‘j’,”jay”); printf(“The value of pi is %f”, 3.1432)

}

## Field-Width Specifier

Field width specifier use to restrict the precision point in flaod/double number

### Example - 7:

int main(){

int event=5; char heat = 'C';

float time = 27.25;

printf("The wining time in heat %c \n", heat);

printf("of event %d was with out field-width is %f\n", event, time); printf("of event %d was with field-width is %.2f", event, time);

}

Write the above program on Turbo C and write output below

**Assignment Tasks**

**Task 01:** Using scanf() and printf(),Create a program with the help of format specifiers to print your marks that include:

Marks of Subject 1 Grad of Subject 1

Marks of Subject 2 Grad of Subject 2

Marks of Subject 3 Grad of Subject 3

**Task02:** Using scanf() and printf()Create a program that has the following output: (the price should be taken as input from the user)

Price of one page: 4.75 Rs

Price of 15 pages: 71.25 Rs Price after discount: 60.00 Rs

**Task 03:** Write a program that can print a table of 9, using scape sequence.

**Task 04:** Attempt example – 7 and note its output.

**Task 05:** Take two integers as input from the user and swap the values.

**Task 06:** Using scanf() and Constant, create a program who takes an input which is radius of circle and calculate area of it.

Area of Circle 𝐴 = 𝜋 𝑟2 where 𝜋 is constant (3.1415)

**Task 07:** Create a program that takes a decimal number from user and displays the whole number part and fraction part of the number separately. The output screen should be as shown below:

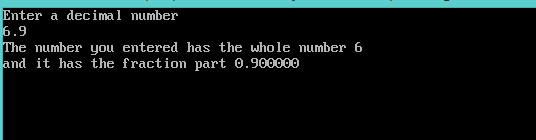


Figure 5 – Output of task 7

**Review Questions**

1. What is variable?
2. Why we need a variable? 3- How a data type works?
3. Why we need a data type?
4. If everything is in binary for the computer then why we need decimal and hexa representations.
5. What are constants?
6. What are the two methods of using a constant value? 8- What are keywords?

9- Define macro? What we mean by a pre-processer directive? 10- Why a macro is used?