**Data Types and C Formatted Input/Output**

**Objectives**

• Use fundamental data types.

• Write simple output statements to display values stored in variables.

• Write simple input statements to read values from the key board.

• Define and use derived data types.

**Variables**

• Variable is a location in memory where a value can be stored for use by a program

• Variables must be declared, before they can be given a value. When declaring a variable, its name and data type should be specified. Every variable has a name, type and a value.

• The declaration allocates the storage location of the appropriate size and associates the name and data type with that location.

**Variable Declaration**

• The format for the declaration of a variable

<data type> < Name of the variable>;

Example:

int quantity;

float price;

double number;

char letter;

**Variable Names**

* Variable name in C is a valid identifier.

An identifier

- can be a series of characters consisting of letters, digits and underscores (\_)

- does not begin with a digit

- may not contain embedded blank spaces

- may not be a reserved word (ex: int, return, if, while, for….)

C is case sensitive (uppercase and lowercase letters are different in C)

i.e. total , Total and TOTAL are three different variable names

**Data types – Examples**

Integers 🡪 462 , -39 , 31285

Real Numbers 🡪 -21.73 , 15.0 , 6.252e-3

Characters 🡪 ‘A’ , ‘@’ , ‘7’ , ‘v’

**Storing values into variables**

• The assignment operation can be used to store a value in a variable or to change the value stored in a variable

• The assignment operator is the equal sign =

• An assignment expression has the form

variable(lvalue) = expression(rvalue)

• It stores the value of the expression (rvalue) into the memory location for the variable (lvalue)

Example:

quantity = 50;

number = 100.5;

amount = quantity \* 55.25;

**C formatted Input and Output**

• All input and output is performed with streams(sequence of bytes)

Input – bytes flow from a device (e.g. keyboard, disk drive) to main memory

Output – bytes flow from main memory to a device (e.g. screen, disk drive)

• Normally standard input stream is connected to the keyboard and the standard output stream is connected to the screen

**Formatting output with printf**

• printf function output data to the standard output stream.

• printf call contains a format control string that describes the output format.

printf( format-control-string, other-arguments);

format control - string describes the output format.

other- arguments correspond to each conversion specification in format-control-string.

Example:

printf( “%d”, 455);

**printf Conversion Specification**

|  |  |  |
| --- | --- | --- |
| Type | Conversion Specification | Description |
| Integer | %d | Display as a signed decimal integer |
|  | %i | Display as a signed decimal integer (d and i are same in printf) |
| Floating-Point | %f | Display floating-point values in fixed-point notation (float or double data type) |
| Character | %c | Display a character |
| String | %s | Display a string |

Example 01 – How to use different conversion specifier in printf

/\* using conversion specifiers in c a program\*/

#include <stdio.h>

int main(void)

{

printf( “%d\n “, 455);

printf( “%d\n “, -455);

printf( “%i\n “, 455);

printf( “%f\n “, 1234567.89);

printf( “%.2f\n “, 3.446);

printf( “%c \n”, ‘A’ );

return 0;

} // end of main function

Output

455

-455

455

1234567.890000

3.45

A

conversion specifier %.2f specifies that a floating point value will be displayed with two digits to the right of the decimal point. If %f is used without specifying the precision, the default precision of 6 is used. When floating values are displayed with precision, the value is rounded to the indicated number of decimal positions for display purposes.

Example 02 – How to display the output of a simple calculation

/\* adding two numbers and display output\*/

#include <stdio.h>

int main(void){

int no1, no2;

int sum;

no1 = 25; // assign value to no1 variable

no2 = 12; // assign value to no2 variable

sum = no1 + no2; // add numbers

printf( "Sum is %d\n", sum); //print sum

return 0;

} // end of main function

Output

Sum is 37

**Reading Formatted Input with scanf**

• scanf function reads from the standard input stream

• scanf contains a format control string that indicates the type of data that should be entered.

scanf( format-control-string, other-arguments);

format control - string describes the input format.

other-arguments are pointers to variables in which the input will be stored.

Example:

int a;

scanf(“%d”, &a);

**scanfConversion Specification**

|  |  |
| --- | --- |
| Conversion Specification | Description |
| %d | Read signed decimal integer. Argument is a pointer to an int |
| %i | Reada signed decimal, octal or hexadecimal integer. Argument is a pointer to an int |
| %f | Reading a floatingpoint value. Argument is a pointer to a float |
| %lf | Reading a floatingpoint value. Argument is a pointer to a double |
| %c | Read a character. Argument is a pointer to a char |
| %s | Read a string. Argument is a pointer to an array of type char |

**C Structures**

• Structures are derived data types.

• They are constructed using objects of other data types.

• Simply, a structure is a collection of related variables under one name

• May contain variables of different types.

**Declaring variables of structure type**

struct employee {

int empNo;

int age;

char gender;

double salary;

} emp1, emp2 ;

emp1 and emp2 are two variables of the structure employee.

**Accessing members of a structure**

• // Input empNo for emp1

scanf(“%d”, &emp1.empNo);

• //print the salary of emp1

printf(“%.2f”, emp1.salary);

• //assign the gender for emp1

emp1.gender = ‘M’;