

9.1 OVERVIEW

Computer does not do anything on its own. The most important skill to learn is how intelligently one can program it, It can be used to solve multidimensional problems. The C being a magic tool for writing programs is used to solve variety of problems; a beginner just need practice and more practice of writing programs to master it.

Here we shall introduce the basic elements of C language. These are actually the building blocks of every C program. The proper use of these basic elements helps a lot to write effective C programs.



* + 1. Identifiers

|  |  |  |
| --- | --- | --- |
| A Textbook o Co | ter Science | Class 12 |

Identifiers are the names used to represent variables, constants, types, functions, and labels in the program. Identifiers in C can contain any number of characters, but only the first 31 are significant to C compiler. There are two types of identifiers in C: standard identifiers and user-defined identifiers.

* + 1. Standard Identifiers

Like reserved words, standard identifiers have special meanings in C, but these can be redefined to use in the program for other purposes, however  this practice is not recommended. If a standard identifier is redefined, C no longer remains able to use it for its original purpose. Examples of standard identifiers include printf and scanf, which are names of input/output functions defined in standard input/output library i.e„ stdio.h. 

9.13 User-defined Identifiers 

In a C program, the programmer may need to access memory locations for storing data and program results. For this purpose memory cells are named that are called user-defined identifiers.

I

C is a Case sensitiVé language. This -means that C compiler consi uppercase and lowercase letters to be distinct characters. For example, the compiler considers SQUARE\_AREA and Square\_Area as two different i$entifiers referring to di(feggpt memory logatiops.

9.2 KEYWORDS

Keywords Or reserved words are the words, which have predefined meaning in C. There are 32 words defined as keywords in C. These have predefined uses and cannot be used or redefined for any other purpose in a C program. They are used by the compiler as an aid to compile the program. They are always written in lower case. A complete list of ANSI C keywords is given at the end of this chapter.

Keywords or reserved words cannot be redefined in the program.

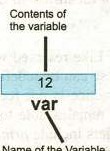
Variables

Variables are named memory locations (memory cells), which are used to store program's input data and its computational results during program execution.

As the name suggests, the value of a variable may change during the program execution. We are familiar with the concept of variable with reference to algebra. The variables are created in memory (RAM); therefore the data is stored in them temporarily. One should not mix the contents of a variable with its address. These are entirely different concepts. The contents of a variable can be thought of as the residents of your neighboring house, while the address of the variable can be thought of as the address of that house (Fig. 9.1).

|  |
| --- |
|  |
|  |
| 12 |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

var



Name

Of

tie

Variable

0002

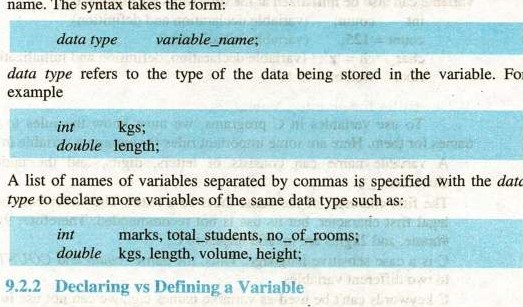
Address of the variable

Memory

9. I : Variable in memory

9.2.1 Declaring a Variable in C

C is a strongly typed language i.e. all variables must be declared before being used. The compiler will report an error if an undeclared variable is used in a program. A variable is declared in C by specifying its type (data type) and



name.

The

syntax

takes

the

form:

For

data

|  |  |  |  |
| --- | --- | --- | --- |
| A Textbook o | ter Science | 147 | Class 12 |

Variable declaration tells the compiler the nan-le Of the variable to be used in the program and the type Of information stored in it. In a C program, the



variable declarations tell the compiler the name of two variables (volume and ch) used to store an integer and character data respectively. 

A variable declaration does not set aside memory location for the data to be stored. It just informs the compiler the name Of the variable and the type of data to be stored in it, while the definition of the variable that set aside memory location for the variable. 

However in C, the variable declaration statement not only declares the variable but also defines it as in case of above two statements. It does not mean that the declaration of a variable can not be separated from its definition in C. C is such a powerful language that provides us all what we need for developing a program. But the discussion is out Of scope of this book. We will learn more on this topic in next classes.

9.2.3 Initializing a Variable 

 Assigning a value to a variable at the time of its declaration is called initializing a variable. In a C program, when a variable is declared, the compiler set aside some memory space for it. This allocated memory space may contain data meaningless to the program (also called garbage). The computations involving this variable may produce unexpected results. To

avoid this situation, all variables should be declared before being used. In C, a variable can also be initialized at the time of its declaration e.g„ int count; (variable declaration and definition) count = 125; (variable initialization) char 'z'; (variable declaration, definition and initialization) float weight = 75.8;

9.2.4 Rules for Naming Variables

* To use variables in C programs, we must know the rules to choose names for them. Here are some important rules for naming a variable in C:  A variable name can consists of letters, digits, and the underscore character ( \_ )
* The first character of the name must be a letter. The underscore is also a legal first character, but its use is not recommended. Therefore, 9winner, #home, and 2kgms are invalid names in C.

C is a case sensitive language. Thus, the names count and COUNT refers to two different variables.

* C keywords can't be used as variable names e.g., we can not use int, void, signed, or while as variable names.
* For many compilers, a C variable name can be up to 31 characters long. (It can actually be longer than that, but the first 31 characters of the name are significant) e.g., Turbo C++ restricts the maximum length of a variable name to 31 characters. Hence, the names (31 characters) and (40 characters) would appear to be the same to the compiler. The compiler does not differentiate these two names because the first 31 characters Of both are the same.  Blank spaces are not allowed in the name e.g„ problem solving is an invalid variable name in C.

A variable can only be declared for one data type.

C programmers commonly use only lowercase letters in variable names, although this isn't required. Using all-uppercase letters is usually reserved for the names of constants.

\_Variable Names should be Readable

Let's consider a pmgram that calculates loan payments could store the value of the prime interest rate in a variable named interest\_rate. The variable name Irlps make its usage clear. We could also create a variable named x or even Ahmed; it doesn't matter to the C compiler. Many naming conventions are used fty variable names. We've seen one style: interest\_rate. Using an Underscore to separate words in a variable name makes it easy to interpret. Another style is to capitalize the first letter of each word. Instead of the variable would be named InterestRate.

9.4 CONSTANTS

A constant is a quantity whose value can not be changed. Unlike a variable, the value stored in a constant can't be changed during program execution. As discussed in chapter 1, the define directive can be used to define constant macros, for example:

\*define PI 3.142857

defines a constant i.e. PI, whose value will remain unchanged during the program execution. C has two types of constants; numeric constants and character constants, each with its own specific uses.

9.4. I Numeric Constants

Numeric constants consist of numbers. It can be further divided into integer and floating point constants. Integer constants represent values that are counted and do not have a fractional part e.g., +56, -678, 8, etc. Floating point constants represent values that are measured e.g., - 4.786, 5.0, 0.45 etc.

9.4.2 Character Constants

|  |  |  |  |
| --- | --- | --- | --- |
| A Textbook o C | Science | 149 | Class 12 |

A character constant is a single alphabet, a single dgit or a single symbol enclosed within apostrophes e.g., 'A' is a valid character constant. e.g.,

'5' etc. The maximum length of a character constant is 1 character. 

9.5 DATA TYPE

In C, the data type defines a set of values and a set 'of operations on those values. We know that computer program manipulates various of data. The data is given to the program as input. The data is processed according to the program instruction and output is returned. In program designing, the data and its type are defined before designing the actual program used to process the data. type of each data value is identified at the beginning of program design. The values or used in a program may be of different types. 

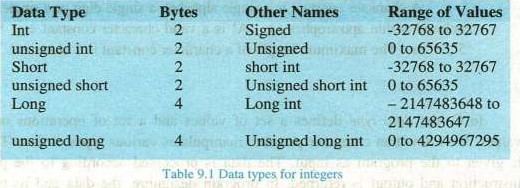
In a C program, We may need to process different type of data. Therefore, variables should be capable of storing data of different types. This is done by associating certain data types to the variables.

 TO work with different types of data, C defines some standard data types and also allows us to define our own data types known as user-define data types. In C, a standard data type is one that is predefined in the language such as int, double, char, float etc. Let' s look at some of the important standard data types:

9.5.1 Data Types for Integers (int, short, long)

 In C, the data type int is used to represent integers — the whole numbers. This means that int variables store number that have no fractional parts such as 1128, 1010, 32432 etc. Along with standard type int, the compiler also supports two more types of integer i.e. short int and long int, often abbreviated to just short and long. In addition to these types, an integer variable can be signed or unsigned. If not mentioned, all integer variables are considered to be signed. The data types signed inf and inr can handle both signed and unsigned whole numbers such as 245, 101(), and -232 etc, where as the data type unsigned can not handle negative numbers.

Because of the limited size of the memory cell, all integers can not be represented by int, short or long. When a variable of type inr is declared, the compiler allocates two bytes of memory to it. Therefore, only the numbers ranging from -21s through 2 1 ' — 1 (i.e. -32768 to 32767) can be represented with the int type variables. The variable of type unsigned int can handle numbers ranging from 0 through 216-1 (i.e. O to 65635). The long is used to represent larger integers. It occupies four bytes of memory and can hold numbers ranging from -231 through 231-1 (i.e. — 2147483648 to 2147483647).



9S.2 Data Typß for moating Point Numbers (float, double, longdouble)

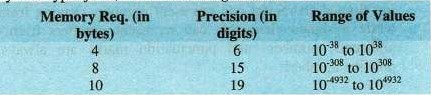
Floating point numbers are the numbers that have a fractional part e.g. 12.35874, 0.54789, and -8.64, 101.003 etc. The floating point numbers are represented in computer in a format that is analogous to scientific notation (floating-point format). The storage area occupied by the number is divided into two sections: the mantissa and the exponent. Mantissa is the value Of the number and the exponent is the power to which it is raised.

For example, in exponential notation the number 245634 would be represented as 2.45634 x 105, where 2.45634 is the mantissa and 5 is exponent. However Tin C, the representation of scientific notation is slightly different e.g. the above number will be represented as 2.45634e5. We don't express the exponent as the power of 10, rather the lettér e or E is used to

n

separate exponent from the mantissa. If the number is smaller than 1 (one),  then exponent would be negative. For example, the number 0.00524 will be represented in computer as 5.24E-3.

ANSI C specifies three floating point types that differ in their memory requirements: float, double, and long double. These data types provide higher precision than the data types used for integers. The following table shows memory requirement, precision and the range of values that can be represented data types float, double and long doublg.

Data Type

float  Double long double

Table 92 data types for floaiing point numbers

9.53 Data Type for Characters — char

The data type charis used to represent a letter, number, or punctuation mark (plus a few Other Symbols). A char type variable occupies 1 byte in memory and can represent individual characters such as 'a', 'x', '5', and '#' etc. (the character '5' is manipulated quite differently than the integer 5 in the computer, so one should not consider both the same. we shall thoroughly discuss the topic in next chapters). In C, a character is expressed as enclosed in apostrophes such as •a', •e', 'o', and 'u' etc.

Like numbers, characters can also be compared, added and subtracted. Let's look at the following program to understand the concept:

#include <stdio.h> void main(void) 

char chl, Ch2, sum; chl = ch2 = •6%, Sum = chl + ch2; printf("Sum = %d", sum);



In fact, characters are stored in the form of ASCII (American Standard Code for Information Interchange) code. When we add, subtract or compare two

152



characters, then instead of characters their ASCII codes are manipulated. In above example. because the ASCII codes of characters '2' and '6' are 50 and 54 respectively therefore the sum is 104. In ASCII, the printable characters have codes from 32 (code for a blank or space) to 126 (code for the symbol The other codes represent nonprintable control characters. The complete list of ASCII charters with their codes is given at the end of this chapter.

The signed and unsigned keywords can be used with char like they can with int. Signed characters can represent numbers ranging from -128 though 127, while unsigned characters can represent numbers from O to 255. English alphabets, numbers and punctuation marks are always represented with positive numbers.

Working "'ith floating point numbers

While working with floating point numbers, we may encounter some problems. For example, manipulation of very large and very small floating point numbers may show unexpected results. When we add a large number and a small number, the larger number may cancel out the smaller number, resulting in a cancellation error e.g. the result of addition of 1970.0 and 0.0000001243 may compute to 1970.000000 on some computers.

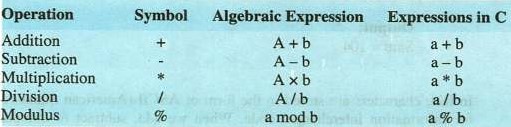
When two very small numbers are manipulated, the result may be too small to be represented accurately, so it will be represented as zero. This phenomenon is called arithmetic underflow. Similarly, manipulation of two very large numbers may result to a too large number to be represented. This phenomenon is call arithmetic overflow.

9.6 OPERATORS C

Operators are symbols, which are used to perform certain operations on data. C is equipped with a rich set of operators. These include arithmetic operators, relational operators, logical operators, bitwise operators, and many others. We shall discuss only the first three of these operators.

9.6.1 Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on values (numbers). The C language incorporates the following standard arithmetic operators:





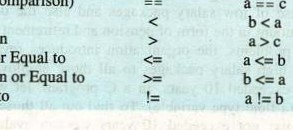
0

The use of first four operators is straightforward. The last operator is modulus (also called remainder operator). Contrary to the division operator, which returns the quotient, it returns the remainder of an integral division. For example, if a, and b are two integers having values 8 and 3 respectively, then the express a % b will be evaluated to 2, which is the remainder of integral division.

9.6.2 Relational Operators

Relational operators are used to compare two values. These operators always evaluates to true or false. They produce a non-zero value (in most cases l) if the relationship evaluates to true and a O if the relationship evaluates tofalse. There are the following six basic relational operators in C:

Suppose a, b and c are three integer variables having values 123, 215 and 123 respectively then:

Operation Symbol Expression Evaluation  Equal to (cornparison) true (non-zero)

Less thanfalse (zero)

Greater thanfalse (zero)

I.zss than or Equal to true (non-zero) Greater than or Equal false (zero)

Not Equal to frue (non-zero)

9.6.3 Logical Operators

Logical operators combine two or more relational expressions to construct compound expressions. The logical operators are && (logical AND), Il (logical OR), and ! (logical NOT).

The first logical operator && (logical AND) when combines two conditions, evaluates to true if both the conditions are ü•ue, otherwise it evaluates to false. The second logical operator Il (logical OR) when combines two conditions, evaluates to true if any one of the conditions evaluates to true, otherwise evaluates to false. Similarly, the third logical operator ! (logical NOT) when applied to a condition, reverse the result of the evaluation of the condition, this means that if the condition evaluates to true, the logical NOT operator evaluates to false and vice versa. The three logical operators can take the following general forms:

expl exp2 exp I Il exp2

!(expression)

Example:

Suppose, if the salary of an employee in an organization is less than Rs. 10,000 and he/she is married than he/she will be given an additional relief allowance. In a C program, let we have two variables; salary (an int type variable and status) a char type variable representing the marital status of the employee. To evaluate this condition, we can use the && operator.

salary < 1000 status 'M' 

Suppose, the organization changes his policy and decides to give relief allowance to all those employees whose salary is less than 5000 or who are married. To find out all those employees who fulfill the criteria, we can evaluate the following condition:

salary < 5000 Il status 'M' 

|  |  |  |
| --- | --- | --- |
| Elements | C | 154 |

In the meanwhile, the organization feels that their employees are facing problems because of low salary packages and also the organization has to spend a huge amount in the form of pension and retirement bonus. Keeping in view all these problems. the organization introduces revised pay scales in which it offers high salary packages to all those employees whose service period has not exceeded 10 years. In a C program, let we have a variable service\_period (a float type variable). To find out all those employees whose service period has not exceeded 10 years we can evaluate the following condition:



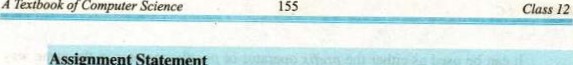
9.6.4 Assignment Operator

In addition to basic C operators (arithmetic, logical, and relational) thete are some other imm)rtant operators, which one should know to write C programs. These includes assignment operator, increment and decrement operators.

The assignment operator is used to store a value or a computational result in a variable. In C, the symbol = represents the assignment operator e.g. in the following statement, values of the two variables, height and width, are multiplied and the result is assigned to the variable Area.

Area = height \* width

The value to the right side Of the operator is assigned to the variable on the left side Of the assignment operator. This statement is also called assignment statement.

Assignment Statement

The assignment statement takes the general form:

variable = expression

The expression on the right side of the operator is evaluated first and the result is then assigned to the variable on the left side of the operator. The expression can be a variable, a constant or arithmetic, relational or logical expression.



Note: Writing variable to the right side and the expression to the left side of assignment operator will cause a syntax error,

9.6.5 Increment and Decrement Operators

The increment operator increases the value of its operand by one. It is denoted by the symbol ++ e.g. count++, where count is a variable. The effect of this expression is equivalent to the following expression:



count count + 1;

When ++ precedes its operand, it is called prefix increment. When the ++ follows its operand, it is called postfix increment. Consider the following statements:



The first statement assigns the value of 10 to the variable j. In the second statement, first the value ofj will be incremented by one (i.e. the value of j will be 11) and then the result will be assigned to the variable i. Hence, the variable i will be assigned the value of eleven (11). Therefore, after execution of the two statements, both the variables will have the same value i.e. eleven (11). Now, consider the following statements: 



The execution of the first statement will take place as in above case. In the second statement, first the value ofj (i.e. 10) will be assigned to the variable i, and then the value of j will be incremented by one. Hence, the variable i will be assigned the value of 10. Therefore, after execution of the two statements, the variable j will have the value of 1 1 and the variable i will have the value of 10.

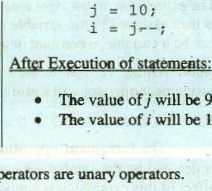
C also provides a decrement operator denoted by the symbol e.g. count-—. The effect of this expression is equivalent to the following expression:

\_gpunv Cpunb.

It can be used as either the prefix operator orpostfix 0Frator in the same way as the increment operator is used. Consider the effect of the following set of

statements:

j = 10;



9

and

10

After Execution of statement>:

* + - The value ofj will be 9 and
    - The value of i will also be 9

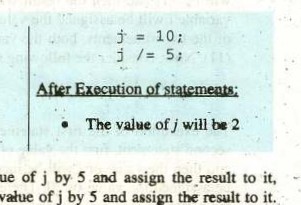
Both increment and decrement operators are unary

The ++ and O}rrators respectively increment and decrement the value of their operand by one. There are four other compound assignment operators that Can increment or decrement the value of their operara by other than one. These are + = , and / = 01krators.

For example, the statement j +2 5 increases the value of j by 5 and the statement — = 5 decreases the of j by 5, Similarly, we also use the operators \* = and / = in the same way.

|  |  |
| --- | --- |
| After | of statemepts; |

* + - The value ofj will be 50



to

statement

The operator multiplies the value of j by 5 and whereas the operator / divide the vNue ofj by 5 assign tkr

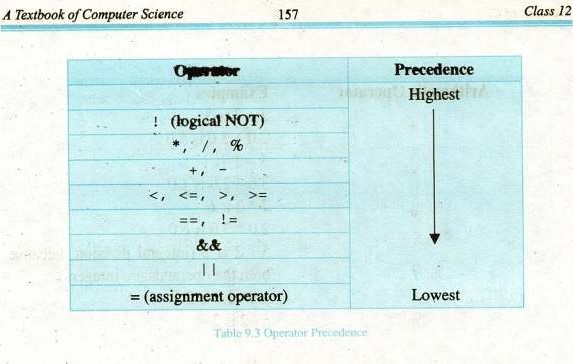
The statement j 5 is equivalent to j = j \* 5 , and the

 / = 5 is equivalent to j =

9.6.6 Opcraiot• erecedence



An operator's precedence determines its or&r of evaluation in an expression. Table 9.3 lists the precedence of all C operators discussed so far, from highest to lowest.



The table shows that the logical NOT orrrator has the highest priority. It is a unary cw•ator — unary operators are those operators that have just one operand. Tlwn comes arithmetic operatcrs, relational operators, logical AND, logical OR and the operNors. which are all binary operators — binary "•eramrs are that have two operands.



tmse

9.7. EXPRESSION

An expression is combination of c»erators and operands. The operand may either be a constant a variik e.g„ a + b, 7 + m etc.

An expression, in which only arithmetic operators operate on operands, is krwwn as arithmetic expression. To solve different mathematical problems, one needs to write uithrmtic expressions. Arithmetic expressions involve integers and floating

Which are manipulated Wiff arithmetic operators,

9.7. I Data Ty pe or an

data type •cf expression (in fact, the data type of the result of expression) the types of its operands. For example, consider the of involving int or doubk type ovwrands is of the form:

operator

If tvth are of int, the result of the expression will be to an iM value. However, in of mixed-type expression, must careful. Ami.xed-tyr expression is in which operands are of diff—t tyßs e.g. if tyi\* of operandl is int and the type of orrrand2 is exßeseion •ill always be to a double type value.

|  |
| --- |
| 3.0 is 5.0 |
|
| 1  1.0 |
| 3.0 is 6.0  2 (integral division, because operands are integers) |
|  |

|  |  |
| --- | --- |
| Arithmetic Operator | Examples  2+3is5  2.0 +  3-2  3.0-2.0is  2\* 3is6  2.0 \* 5 / 2 both  the  5/2is2 |
|
|
|
|  |

is

is

5.0/ 2.0is2.5

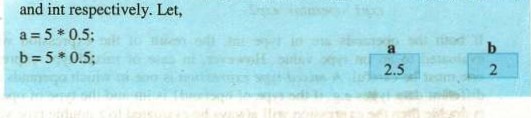
Working with Division Operator

|  |  |  |  |
| --- | --- | --- | --- |
| Elements ofC | 158 | C | rer9 |

The manipulation of division operation is slightly different from other arithmetic operations in C. One should be careful while dividing numbers, as the result of division of two integers may not be an integer. C handles the division intelligently.

When the divisor and the dividend both are integers, the fractional part of the quotient (if exists) is truncated. For example the value of 7.0/2.0 is 3.5 but the value of 7/2 is the integral part of the result i.e. 3. Similarly, the value of 198.0/100.0 is 1.98, but the value of 198/100 is the integral part only i.e. 1. That's how C performs the division operation. So, to get the accurate result, at least one floating point number should be involved in division operation. Otherwise, integral division will take place and we will get the integral value.



Note: When a type double expression or value is assigned to a type int variable, the fractional part is truncated since it can not be represented in int type variable. For example, let a and b be two variables of type double and int respectively. Let,

of 159 Class

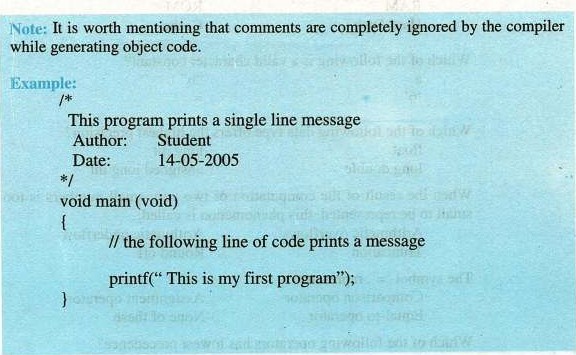
9.8 Comments

Comments are used to increase the readability of the program. With comments, informative notes are inserted in the program's code, which helps in debugging and modifying the program. In C, there are two ways to comment the code; one can insert single line comments by typing two (forward) slashes at the start ofthe note such as:

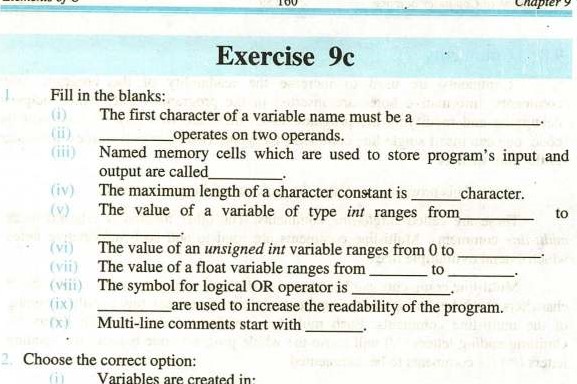
// This program calculates the factorial of the given number

These are called single-line comments. The other method is referred to as multi-line comments. Multi-line comments are used to add such informative notes which extend to multiple lines. 

Multi-line comments can be inserted to the code (program code) by placing /\* characters at the beginning of the comments (informative note), this is called opening of the multi-line comments. Each multi-line comment must end with letters Omitting ending letters (\*/) will cause the whole program code beneath the opening letters (P) for comments to be commented.



Elements OfC



9

Variables

are

created

in:

Hard Disk 

 Which of the following is a valid character constant?

"b"

 Which of the following data type offers the highest precision? float long int long double unsigned long int

When the result of the computation of two very small numbers is too small to be represented, this phenomenon is called;

|  |  |  |
| --- | --- | --- |
|  | Arithmetic overflows | b) Arithmetic underflow |
|  | Truncation  The symbol , represents: | Round off |

Comparison operatorAssignment operator

Equal-to operatorNone of these

 Which of the following operators has lowest precedence?

a)

 Relational operators are used to:

Establish a relationship among variables

Compare two values



Class

1. Construct compound condition 
2. Perform arithmetic operations
3. C is a strongly typed language, this means that:
   1. Every program must be compiled before execution
   2. Every variable must be declared before it is being used
   3. The variable declaration also defines the variable
   4. Sufficient data types are available to manipulate each type of data
4. The logical not operator, denoted by 2, is a:

a) Ternary operator b) Uniary operator

c) Binary operator Bitwise operator

b is equivalent to:

b)

d)

1. Write T for true and F for false statement.

printf and seanf are standard identifiers.

In C language, all variables must be declared before being used.

(iii) Standard data types are not predefined in C language.

The double data type required 4 bytes in memory.

In Scientific notation, the exponent represents the value Of the number and mantissa represents the power to which it is raised.

* 1. The symbol for modulus operator is
  2. The symbol is used to compare two values.
  3. Operator precedence determines the order of evaluation of operators in an expression. 

For many compilers a C variable name can be up to 31 characters. C program can only use lowercase letters in variable names.

1. What is an identifier? Discuss the two types of identifiers in C.
2. What is a variable? Discuss the difference between declaring and defining a variable.
3. Write down rules for naming variables in C.

Differentiate the following:

* 1. Constant and variable
  2. Character constant and Numeric constant
  3. Standard data type and User defined data type



* 1. Keyword and Identifier

1. What is a data type? Discuss various C data types to manipulate integers, floating point numbers and characters.



1. How many types of operators are available in C? Describe briefly. Also mention their precedence.



1. What data type would you use to represent the following items: number of children at your school, a letter grade on an exam, and the average marks Of your class?
2. Which of the following are valid variable names in C?

 total marks

* 1. double (iv) average-score  room#

(vi) 

* 1. no\_of\_students
  2. long

(ix)

MAX\_SPEED

1. Write a note on the following:

Arithmetic Expression

Comments in C

13, Let w, x, y, and z be the name of four type float variables, and let a, b and c be the names of three type int variables. Each of the following statements contains onc or more violations of the rules for forming arithmetic expressions. Rewrite these statements so that it is consistent with these rules. y;



4.0

w\*

* 1. a = 6b4;
  2. c = 3(a + b);  z = 7w + xy;

14. Assume that you have the following variable declarations:

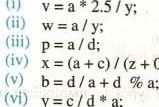
int a, b, e, d, b; float v, w, x, y, z ;



Evaluate each of the following statements assuming is 2 , Z is —

1. 3, c is 1, d is 3, y is 0.3E+1.

(z +0.3);



(i)

y=c/d

\*

a;

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  | Chåpter  10 | |

10.1 OVERVIEW

In previous chapters, we have studied the basics of C programs. This lesson covers basic input and output features of C language. Usually input and output form an important part of any program. To be nore interactive, a program needs to able to accept data and show results.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | er Science |  | Class 12 |

In C, the standard input/output library provides functions to perform input and output operations. By standard input and output, we nwan the keyboard and monitor respectively. In C, these input/output operations are performed by two standard input/output functions, these are printf( ) and scanf( These functions can be accessed by including the standard input/output library (stdio.h) in the program. Let us have an overview of standard input/output functions.

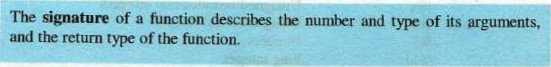
10.1.1 printf Function

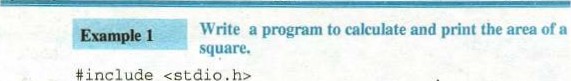
 To see results of program execution, we must have way to speci$' what variables values should be displayed. The standard library function printf (pronounced as print-em is used for formatted output. It takes as arguments a format string and an optional list of variables to output. The values of. variables are displayed according to the specifications in the format string.

The printf ( ) function will take the form:

printf(format string, varl, var2, var3, string);

The fonnat string is a character string — nothing more and the variables are optional. The easiest way to understand this is by example.





#

include

<stdio

In

ut/Out

void

main

(

)

int height, width, area; height = 5 width 4 ; area height \* width; printf ("Area of Square = \*d", area) ;

Here's the output of the program:

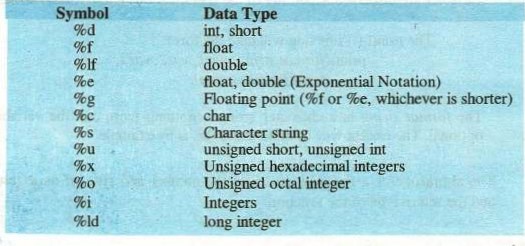
4rea of Square 20

In the above program, the first line is the variable declaration staternent. In second and third lines, values are assigned to the variables height and width. Fourth line of code descritrs the arithmetic expression for calculating the area of the square and the •result is assigned to the variable area. Fifth and the last line of code is the printf( ) statement, which displays result on the screen. In case of printf, the first pararneter is always a string (e.g., "Area of Square"), which should be enclosed in double quotes. This string is called the format  string. Format string may include any number of format specifiers such as %d. The list of variables separated by commas, whose values are to be displayed in the result, will follow the format string.

10.1.2 Format Specifier

Forrnat slrcifiers specify format in which value of a variable should displayed on screen. Format Sl\*cifiers are specified in the forrnat string.

For instance, in the above program the printf( ) staternent contains the symbol %d, which is format specifier for the variable area. For different types of variables, different format specifiers are used. Here is the list Of format specifiers:





Example 2 Write a program that adds two floating point numbers and shows their sum on the screen.

# include void main (void) float varl, var2, res ;

varl - 24.27; var2 - — 41. 50; res varl , var2 , res) ;



varl

+

var2;

Here's the output of program

24.27 +41#65.7T 

10.1.3 Field-width Specifier



Class

12

In a C program, the of columns used to display a value on the screen is referred to as field-width. Field-width specifiers the numt.rr of columns that should be used to print a value Formatting Integers

We simply need to add a number between the % and d of the %d format specifier in the printf format string. This number specifies the field-width or the number of columns to be used for the display of the value. The statenrnt printf("Area = %4d", area); indicates that four colurnns will be used to display the value of area. Suppose the value of the variable area is 25. Two extra spaces will be padded before 25 on the screen to complete the length of 4. output of the above statement will be as follows:

Area = CL125

Here, Cl represents a blank space. This space will not be displayed as a printed character in actual output. In this way the value of 25, which requires two spaces to be displayed, will occupy four spaces (columns) on the screen. The reason is that the format specifier for area (%4d) allows spaces for four digits to be printed. Because the value of area is 25, therefore its two digits are right justified, preceded by two blank spaces.

The following table shows how values are displayed using different format



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| VallE | Format  Specifier | Displayed Output | Value | Fornnt  Specifier | DÉplayed Output |

0786 -786 -786 

786 11786 -7860-786

786 TL786 -786 DO-786

786 786 -786 -786



The last row of this table shows that C expands the field width if it is too small for the integer value displayed.

Formatting Floating Point Numbers

For format specification of floating point numbers, we must indicate both the total field width needed and the number of decimal places desired. The total field width should be large enough to accommodate all digits before and after the decimal point e.g„ to display 15.245 and 0.12 the total field width should be six and four respectively. It should noted that for nurntrrs smaller than zero, a zero is always printed Irfore the decimal point. Therefore the total field width should include a space for the decimal point as well as for the minus sign ifthe numtrr can be negative.

The general form for the format specifier for a floating point value will %m.nf, where m represents the total field width, and n represents the desired number of decimal place. For instance, the statement

printf("Height = 6.2f", height);

indicates that the total field width for the value of the variable height is 6, and the accuracy is of two decimal places. The value of height will be rounded off to two decimal places and will be displayed right justified in 6 columns. While being rounded off, if the third digit of the value's fractional part is 5 or greater, the second digit is increased by one othenvise the third digit is



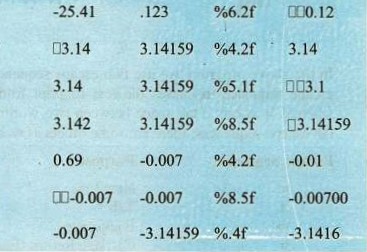
\*emember: A format always\_begins with thesymbol€



The following table shows how values are displayed using different format



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Value | Format  Specifier | Displayed Output | Value | mai misplayed  Specifier Output | |  |

 -25.41 %6.2f

3.14159 %5.2f

3.14159 %3.2f

3.14159 %5.3f

.6789 %4.2f

-,007 %8.3f

-.007 %.3f

10.1.3 Escape Sequences

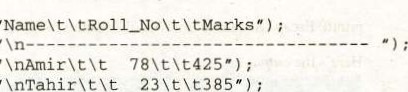
Escape sequences are characters which are specified in the format string of the printfstatement in combination with a backslash These cause an cscapc from the normal interpretation of a string so that the next character is recognized as having a special meaning. For example, consider the following program

Remember: Escape sequence characters always begins witha backslash

Example 3 Write a program that will demonstrate the use of escape sequences. #include <stdio.

void main (void)

printf ( printf ( " printf ( printf (

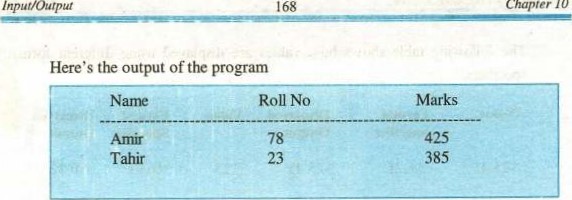


23

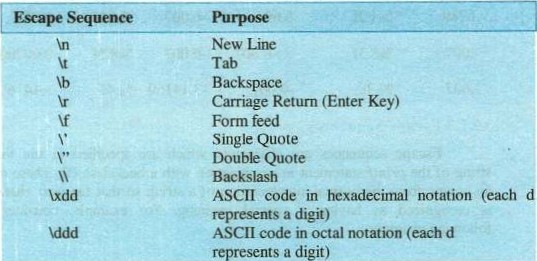
t

\

t385"



In the above program, we use two escape sequences. These are it and The escape sequence causes the text to print from the start of the next line, whereas it inserts a tab space between two words. In addition to newline and tab escape sequences; there are some others as well. Here is a list of them:

represents a digit) ASCII code in octal notation (each d represents-a digit)

We have discussed the purpose of first two escape sequences. The escape sequence causes the cursor to move one space left, the form-feed (if) rmves to the next page on printer. It is important to note that one can not display a single or double quote on the screen without using the escape sequences V' and V'. The reason is that the format string of the printf function is enclosed in a double quote. When a double quote is specified in the format string, it is treated as the closing double quote. That's why, single and double quotes are always written with backslash. For example the statement printf('Escape Sequence is a feature of

Here'S the output



|  |  |
| --- | --- |
| Textbook Of | 12 |

r

10.2 SCANF FUNCTION

Most of the programs are interactive in nature. Till now, we did not learn a way to write interactive programs. C is featured with a range of functions to accept user input in variety of forms. The scanf (pronounced as scan-em function is versatile as it is equally good for numeric as well as string input.

It takes as arguments a format string and a list of variables to hold the input values. Here is the syntax Ofscanf Rinction:

scan(format string, &varl, &var2, &var3,

Let us consider the following program to understand the working of scanf function:

Write a program to convert the distance in kilometers into meters. #include <stdio .

void main (void)

double meter, kilometer ;



// prompt the user to enter kilometers printf ("Enter distance in kilometers ;

/ / take input scanf ( "%lf", &kilometer) ; meter = kilometer printf ( kilometers =  meters" , kilometer, meter) ;

Here's the sample output of the program



In the above program, first line is the declaration of variables meter and kilometer. The next executable statement is printf, which displays a message for the user to enter distance in kilometers. The next is the scanf statement. When the program reaches this line of code, the flow of execution stops until the user enters a value. The format string of scanf is "0/01?' which tells the scanf what kind of data to copy into variable kilometer. The forrnat string of scanf consists of a list Of R)rmat specifiers only; no other value or text can be specified in it.

Instead of the variable name, the scanf requires address of the variable to the input value into it.

170



Notice that in the call to scanf, the name of variable kilometer is preceded by an ampersand character In C, & is actually the address of operator. In scailf, the address of operator (&) tells the scanf funciton the address of the variable where the input value is to store. If & is omitted, the scanf will not be able to locate the variable in memory, hence it will be unable to store the value into the variable and the program will find a garbage value in the variable.

number entered

40.5

|  |  |
| --- | --- |
|  |  |
| 40.5 |  |

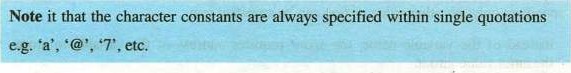
fig. 10.1 Input value is stored in variable

10.3 CHARACTER INPUT

In C, there are many functions to accept character input. The versatile scan/ can also be used for this purpose. But scanfrequires pressing the return key at the end of input value. In some cases, it is desirable to input characters without pressing the return key. For example, in a game while controlling the movement of a space ship through arrow keys we can't afford to press return key each time after pressing an arrow key. To overcome such situations, C is equipped with many other functions specialized for character input. getch and getche are examples of such functions. These are part of the conio (console input output) library.

103.1 getch and getche Functions

The getch and getche functions are very handy in character manipulation. In  contrast to the getch function which does not echo the character typed, the getche function echo the typed character. Both of these functions do not accept any argument. However they return the typed character to the calling function. One does not need to press the return key (ENTER key) after typing the character. The moment a character is typed, it is imidiately returned by the function to the calling module.





Class

12

Textbook

Of

Computer

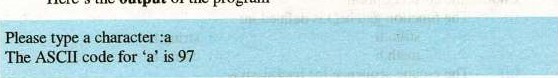
Example f Write typed by a progeam the user.that displays the ASCII code Of the character

#incLude # include void main ( ) char ch;

printf( "please type a character 2") ; ch getche( ) ;

ASCII code for V is ed" ch. • eh);

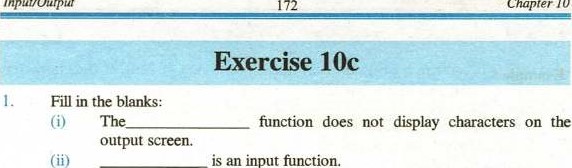
Here's the output of the program



Note it that, in this program we include a new header file conio.h. This file contains the definition of functions getch() and getche(). In this program, the statenwnt ch = getche(); can replaced with the statenrnt ch = getch();

In the later case, the typed character will not shown on the screen and the output will be as follows:

When the 3rd line of the program is executed, it waits for a character to be typed. As soon as a character is typed, the very next line executes imidiately without waiting for the return key to be typed. And the function getche() returns the typed character to the main function, where it is assigned to the variable ch.



an

input

1. %x is a format specifier for
2. Escape sequences always with a

The printf function is defined in

1. The ASCII code for Escape key is
2. The escape sequence represents the carriage return.
3. There are total columns on the output screen.
4. The symbol for address of operator is \dddd is used to print ASCII code in

|  |  |  |
| --- | --- | --- |
| 2. | Choose the correct option:  (i) The function getche() is defined in: |  |
|  | a) stdio.h b) | string.h |
|  | c) math.h d)     The escape Sequence for backslash is: | conio.h |

 a) b)

c) d)

|  |  |  |  |
| --- | --- | --- | --- |
| (iii) | The format specifier %u is used for: |  |  |
|  | a) integer | b) | unsigned short |
| (iv) | c) unsigned float | d) | unsigned long int |
|  | a) Arithnrtic overflows | b) | Arithnrtic underflow |
|  | c) Truncation  The symbol , represents: | d) | Round off |
|  | Comparison operator | b) | Assignnrnt operator |
|  | c) Equal-to operator | d) | None of these |

Which Of the following operators has lowest precedence?

a) 

c)

1. Relational operators are used to:
   1. Establish a relationship among variables

h) Compare two values

* 1. Construct compound condition
  2. Perform arithrnetic operations

1. C is a strongly typed language, this nrans that:
   1. Every program must be compiled before execution
   2. Every variable must be declared Irfore it is being used
   3. The variable declaration also defines the variable



Class

n

* 1. Sufficient data types are available to manipulate each type of data (ix) The logical not operator, denoted by 2, is a:

a) Ternary operator b) Uniary operator



c) Binary operator d) Bitwise operator (x) a b is equivalent to:

a)

c) d) b=b+a

1. Write T for true and F for false statenrnt:

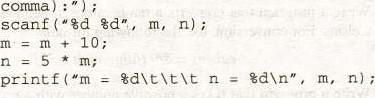


(i) printf and scanf are standard identifiers.

 In C language you must declare all variables before using them. 

 (iii) Standard data types are not predefined in C language. (iv) The double data type required 4 bytes in rnemory.

1. What do we mean by standard input and output? Illustrate the use Of printf() and scanf() functions. 
2. Illustrate the difference between format specifiers and field-width specifiers with examples
3. Define the term 'escape sequence'. List nanrs and uses of any five escaF sequences. 7. 
   1. ShO\*•• the output displayed by the following program the data entered are 10 and 15. \*include void main ( )

int m, n; printf "Enter two numbers (separated by cornma) : ;

* 1. Show the contents Of mernory (for variables a and ) before and aner the execution Of the above program.
  2. Write the program in example 5 using scanf function.

8.

* 1. Show how the value -17.246 would be printed using the formats %8.4f, %8.3f, %8.2f, %8.1f, %8.0f, and %0.2f.
  2. Assuming x (type double) is 21.335 and y (type int) is 200, show the output of the following statements (on paper). For clarity, use the symbol L] to denote a blank space.

 x, y) ;

* 1. If the variables a, b, and c are 307, 408.558 and -12.31, respectively, write a staternent tha t will display the following line: (for clarity, the symbol Cl shows a blank space)

003070010408.561103-12.3

1. Write a program that asks the user to enter the radius of a circle and then computes and displays the circle's area. Use the formula

area = PI x radius x radius

where PI is the constant value of 3.14159. (Note: Define a constant macro PI with \*define directive)

1. Write a program that stores the values 3.456E10 and 50 in separate  memory cells. Your program should get the first three values as input data, but use an assignment statenrnt to store the last value.

H. Write a program that converts a temperature in degrees Fahrenheit to degrees Celsius. For conversion, use the following formula celsius = 5/9 ffahrenheit — 32)

12. Write a program that takes a positive number with a fractional part and rounds it to two decimal places. For example, 25.4851 would round to 25.49, and 62.4431 would round to 32.44,

|  |  |
| --- | --- |
|  | Chapter  11 |

11.1 OVERVIEW

While writing C programs, the programrner may need to choose a path to execute one or more statements through the program based on a certain criterion; Decision constructs provide a way to neet this requirenrnt. This chapter introduces various selection structures available in C, which can be used in such situations.

11.1.1 Control Structures

Control structures are staterrrnts ued to control tie flow of execution in a program or function. C control structures enable us to group individual instructions into a •ingle logical unit with one entry point and one exit point.

Program instructions can be organized into three kinds of control structures to control execution flow i.e. sequence, selection and repetition. All programs, whether simple or complex, use these control structures to implenrnt the program logic.

|  |  |  |
| --- | --- | --- |
| A Textbook co | ter Science | Class 12 |

Until now we have using only sequential flow, which is also called default flow. In case of sequence structure, instructions are executed in the sarne order in which they are specified in the program. A compound statement refers to a group of statements enclosed in opening and closing braces such as:

statement} ; statement2 ;

Statement,t ;

Control flows from statement/ to statement„ in a logical sequence. Here We shall discuss the selection structure in detail, and in next chapter the repetition structure will be discussed. Solutions of sorne problems require steps with two or more alternative course of action. A selection structure chooses which staternent or a block of statements is to execute. In C, there are two basic selection statenrnts:

* if—else
* switch

There are some variations of if — else structure which will be discussed here. The third control structure is repetition, which is also called iteration or loop. It is a control structure, which repeats a statenrnt or a group of statements ina program. C provides different types of loop structures to used in various situations. These include for loop, while loop, and do-while loop.

11.2 IF STATEMENT

if is one of the keywords in C language. It is used to select a path flow in a program based on a condition. A condition is an expression that either evaluates to true (usually represented by I) or false (represented by O). The result of this evaluation can be assigned to a variable. For example, consider the following program:

\*include void main ( ) int age, status;

|  |  |  |
| --- | --- | --- |
| 176 | Ch | ter 11 |

printi ( "Enter the age scanf &age) ;

status = (age > 60) ; td", status) ;

|  |  |  |
| --- | --- | --- |
| The value of the variable status will | 1 if | age is greater 6(), otherwise |

status will be O. Here's the output of the program:

Case 1 (When age is less than 60) Case 2 (When age is greater than 60)

Enter the age :45 Enter the age

Status = O Status = 1

11.2.1 Simple if Statement

if statenrnt is the simplest form of decision constructs. It allows a staternent or a set of staternents to be executed conditionally. The general form of simple if staterrnt is: if (condition)



statement, ; statement2 ;



The statement(s) in the block of if statement are executed if the Condition is true; otherwise these are skipped. If there are more statements in if block then these should be enclosed in braces as a cornpound statenrnt. However, in case of a single statement the braces are optional.

Flow char' of simple i/ statement



Class

12

o

ter

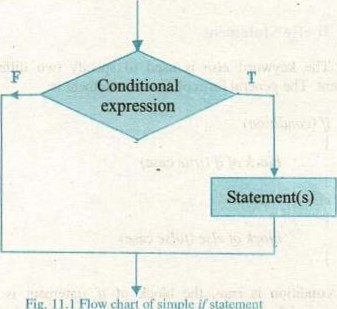


Fig.

This flow chart describes a situation where a simple ifstatement can be used.

Example 1 A program that calculates the square root of a given  number

\*include

# include void main (void)

double x = 0.0, square\_root — 0 0 prir.tf ("Enter a number ;



if (X > O) square\_root: sqrt (x) ;

As square root of negative is imaginary, therefore the program finds the square root of positive numbers only. In this program we have used math library. This includes the definition of sqrt and rnany other nuthematical functions. We shall discuss it later in this book.

A flow chart is the pictorial representation of a program.

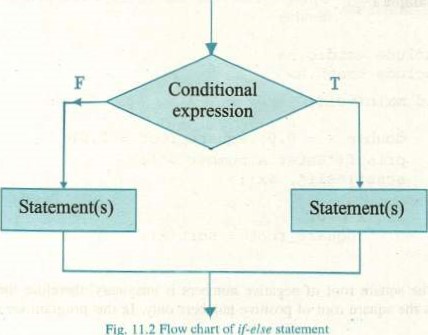
11.2.2 If-else Statement 

The keyword else is used to specify' two different choices with if statenrnt. The general form of if-else statenrnt is:

if (condition) block of if (true case)

block ofelse (false case)

If the condition is true, the block of if staterrrnt is executed and if the condition is false, the block of else statenrnt is executed. The following flow chart may help to explain the idea.



Fig,

.2

Flow

chart

of

(f•e/se

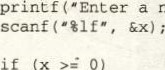
Statement

The program in example 1 can be written using if-else statenrnt as follows:

#incLude

# include void main(void)

double x = 0.0, square\_root 0 0 a number ;



if

(X

O)

square\_root = sqrt (x) ; printf ("Square root of — square\_root) ;

else printf ("Square root can not be calculated") ;



ter

There are two blocks of statenrnts in this program, either of which is conditionally executed. If the condition evaluates to true the square root of x will be calculated and the output will be shown on the screen and in case the condition evaluates to false, the message "Square root can't be calculated" will be displayed.

It is very important to note that the block of if is enclosed in braces, and no bracket has been used in the body of else staterrrnt. The reason is that we want to execute multiple statenwnts in case of true condition, so these must represented as a compound statenrnt. If we omit the braces from the body of ifstatement, the following error nrssage will appear:

Illegal el se without matching i f

So, to avoid this nessage, one should always enclose a compound statement in braces. If there is a compound statenrnt in the body of else, omitting braces will not cause the above-nrntioned error, rather only one statenrnt after the else will be treated as the body of else and the rest of the statenEnts will always be executed sequentially.