Chapter # 09
Elements of C Language

Q. What is an identifier? Discuss the two types of identifiers in C?

The identifiers are the names used to represent variable, constants, types, functions and labels in the program. Identifier is an important feature of all computer languages. A good identifier name should be descriptive but short. 10-12 characters written

An identifier in C language may consist of 31 characters. If the name of an identifier is longer than 31 characters, the first 31 characters will be used. The remaining characters will be ignored by C compiler.

Rules for Identifier

Some important rules for identifier name are as follows:

- The first character must be an alphabet or underscore (_). a,b,c 4-6
- The identifier name must consist of only alphabetic characters, digits or underscores. abc 1,2,3 (_)
- The reserved word cannot be used as identifier name.

Types of Identifiers

There are two types of identifiers in C language:

- Standard Identifiers
- User-defined Identifiers

1. Standard Identifiers

A type of identifier that has special meaning in C language is known as **standard identifier**. C cannot use a standard identifier for its original purpose if it is redefined.

Example output input

printf and scanf are examples of standard identifiers. These are the names of input/output functions defined in standard input/output library stdio.h.

2. User-defined Identifiers

A type of identifier that is defined by the programmer to access memory location is known as **user-defined identifier**. The user-defined identifiers are used to store data and program results.

Examples

Some examples of user-defined identifiers are **a, marks** and **age** etc.

Q. What do you know about keywords of C language?

Keywords

Keyword is a word in C language that has a predefined meaning and purpose. The meaning and purpose of a keyword is defined by the developer of the language. It cannot be changed or redefined by the user. Keyword can be used for the same purpose for which it is defined. They help the compiler in compilation process. Keywords are written in lowercase.

Keywords are also known as reserved words. There are different types of keywords in C language. The total number of keywords is 32.

List of Keywords

Following is a complete list of keywords:

auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	float	return	typedef	
default	for	short	union	

Figure 9.1: Keywords of C language

Q. What is a variable? Explain its purpose.

A variable is a named memory location or memory cell. It is used to store program's input data and its computational results during execution. The value of a variable may change during the execution of program. However, the name of variable cannot be changed.

The variables are created in RAM. RAM is a temporary memory. That is why the data stored in variables is also temporary. It can only be used and processed during the execution of program. The data stored in the variable is automatically removed when program ends.

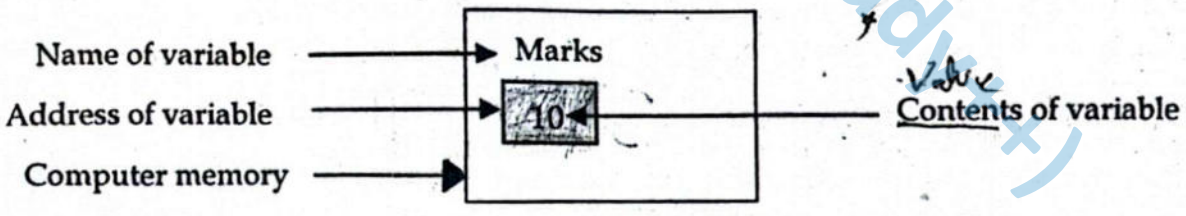

Figure 9.2: Variable in memory

In the above figure:

Name of variable	It refers to an identifier that represents a memory location.
Address of variable	It refers to the memory location of the variable.
Contents of variable	It refers to the value stored in memory location referred by variable.

Q. What is a variable declaration? Explain with examples.

Variable Declaration

The process of specifying the variable name and its type is called **variable declaration**. A program can have as many variables as needed. C is a **strongly typed** language. It means that all variables must be declared before they are used in the program. A variable can be declared anywhere in the program before its use.

The variable declaration provides information to the compiler about the variable. The compiler uses the declaration to determine how much memory is needed for each variable. Different types of data require different amount of memory. The required number of bytes are allocated when a variable is declared. For example, an integer variable requires 2 bytes whereas a character variable requires 1 byte. The memory bytes are used to store the value of the variable.

Once a variable is declared, its data type cannot be changed during program execution. However, the value of variable can be changed during execution.

Syntax

The syntax of declaring variables is as follows:

```
data_type variable_name;
```

data_type It indicates type of data that can be stored in variable.

variable_name It refers to the memory location of the variable.

Examples

Different types of variables are declared as follows:

```
int marks;
float average;
char grade;
double salary;
```

In the above examples, **int**, **float**, **char** and **double** are keywords that indicate data types. The words **marks**, **average**, **grade** and **salary** are variable names.

Many variables of same data type can also be declared in single line. In this case, each variable name is separated by comma as follows:

```
int a, b, c;
```

Q. What is variable initialization? Explain with examples.

The process of assigning a value to a variable at the time of declaration is known as **variable initialization**. The equal sign = is used to initialize a variable. Variable name is written on left side and the value is written on the right side of equal sign.

The compiler automatically allocates the required memory for the variable when it is declared. The memory location may already contain some data that is meaningless for the program. This meaningless data is known as **garbage value**. It may produce unexpected results in some computations. All variables should be initialized to avoid this problem.

Syntax

The syntax of initializing a variable is as follows:

```
type_name variable = value;
```

type_name	It indicates the data type of the variable to be initialized.
variable	It is name of the variable to be initialized.
=	It is the assignment operator used to initialize a variable.
value	It is the value to initialize a variable.

Example

Some examples of variable initialization are as follows:

```
int n = 100;
int x = 50, y = 75;
float average = 50.73;
char grade = 'A';
```

Q. Discuss the difference between declaring and defining a variable.

Generally, there is a slight difference between declaring and defining a variable. The **variable declaration** specifies the name and type of the variable to the compiler. It does not allocate memory location for the data to be stored in the variable.

The **variable definition** specifies the name and type of the variable as well as allocates the memory location to store the data in the variable.

In C language, variable declaration and variable definition are not specified separately. The variable definition is automatically performed when a variable is declared. It means that variable declaration also allocates the memory required by the variable to store data.

Q. Write down the rules for naming variables in C language with examples.

Following are some rules for naming variables in C language:

1. Variable may include letters, numbers and underscore (_).
2. The first character of variable must be a letter or underscore _. The use of underscore is not recommended. The variables 9minute, #home and 2kg are invalid.
3. Blank spaces are not allowed in variable names. The variables my var and your car are invalid.
4. Both upper and lower cases are allowed. A user-defined variable is conventionally written in lower case. The constants are conventionally written in upper case.
5. Special symbols cannot be used as variable name.
6. Reserved word cannot be used as variable name. The names int, void and while are invalid variables.
7. A variable can be up to 31 characters long for many compilers. If a variable consists of more than 31 characters, only first 31 characters will be used. The remaining characters will be ignored.
8. A variable can be declared only for one data type.

The variable name must be meaningful and readable. It helps the user to understand the purpose of the variable. For example, a variable to store the marks of student should be

Marks. If a variable consists of multiple words, first letter of each word should be capitalized to increases readability. A variable **MyPhoneNo** is more readable than **myphoneno**.

Examples

Some examples of valid variables are as follows:

Marks sub1 _test f_name

Q. What is constant? Explain different types of constants.

Constant

A constant is a quantity that cannot be changed during program execution.

Types of Constants

C language provides following types of constants:

- Numeric constants
- Character constants
- String constant

1. Numeric Constants

Numeric constant consists of numbers. It can be further divided into two types:

a) Integer Constant

Integer constants are numeric values without fraction or decimal point. Integer constants represent values that are counted. Both positive and negative integer constants are used in C programs. The minus sign - is used for negative integer constants. If no sign is used, the value is positive by default.

Examples

Some examples of integer constants are as follows:

87 45 -10 -5

b) Floating Point Constants

Floating point constants are numeric values with fraction or decimal point. Floating point constants represent values that are measured. Both positive and negative floating point constants are used in C programs. The minus sign - is used for negative floating point constants. If no sign is used, the value is positive by default.

Examples

Some examples of floating point constants are as follows:

50.75 10.22 –13.42

2. Character Constants

Any character written within single quotation mark is known as character constant. All alphabetic characters, digits and special symbols can be used as character constants. The maximum length of a character constant is 1 character.

Examples

Some examples of character constants are as follows:

'A' 'n' '9' '=' '$'

3. String Constants

A collection of characters written in double quotations mark is called string or string constant. It may consist of any alphabetic characters, digits and special symbols.

Examples

Some examples of sting constants are as follows:

"Pakistan" "123" "99-Mall Road, Lahore"

Q. Differentiate between Variable and Constant.

The purpose of variables and constants is very different. The main difference between a variable and constant is as follows:

	Variable	Constant
Definition	It is a named memory location.	It is a named value.
Use	Its value can change while a program is running.	Its value always remains same while a program is running.

Q. What is data type? List different categories of data types in C language.

Data Types

The data type defines a set of values and a set of operations on those values. The computer manipulates various types of data. The data is given to the program as input. The data is processed according to the program instructions to produce output. The data and its type are defined before designing the actual program that is used to process the data. The type of each data value is identified at the beginning of program design.

C program may need to process different types of data. Each data type require different amount of memory. C language provides the following data types:

Data type	Purpose
int	To store numeric values
float	To store real values
double	To store large real values
char	To store character values

Table 9.1: Data types in C language

Categories of Data Types

C language provides two ways to use data types:

1. Standard Data Type

C language defines some standard data types. A data type that is predefined in the language is called standard data type. Some examples of standard data type are int, float, long and char etc.

2. User-defined Data Type

C also allows the user to define his own data types known as user-defined data type.

Q. What is integer data? Describe different data types to store integer data.

Integer data is the numeric value with no decimal point or fraction. It includes both positive and negative values. The minus sign – is used to indicate negative value. If no sign is used, the value is positive by default.

Examples

Some examples of integer values are 10, 520 and -20.

Types of Integer Data

C provides different types of integer data. These are as follows:

1. int
2. short int
3. long int
4. unsigned int
5. unsigned long int

1. int Data Type

int data type is used to represent integer values. An integer variable stores a number with no fractional part such as 11, 10 and 25 etc. It takes two or four bytes in memory depending on the computer and compiler being used. In MS-DOS, it takes two bytes and its range is from -32768 to 32767.

2. short Data Type

short data type is used to store integer values. It takes two bytes in memory. Its range is from -2^{15} to $2^{15}-1$. It means that the valid value is from –32768 to 32767. The short data type is also called short int.

3. unsigned int Data Type

unsigned int data type is used to store only positive integer values. It takes two bytes in memory. Its range is from 0 to $2^{16}-1$. It means that the valid value is from 0 to 65,535. The unsigned int is also called unsigned.

4. long Data Type

long data type is used to store large integer values. It takes four bytes in memory. Its range is from –2,147,483,648 to 2,147,483,647. The long is also called long int.

5. unsigned long Data Type

unsigned long data type is used to store large positive integer values. It takes 4 bytes in memory. Its range is from 0 to 4,294,967,295. The unsigned long is also called unsigned long int.

Data Type	Size in Bytes	Description
int	2	Ranges from –32,768 to 32,767.
short	2	Ranges from –32,768 to 32,767.
unsigned int	2	Ranges from 0 to 65,535.
long	4	Ranges from –2,147,483,648 to 2,147,483,647.
unsigned long	4	Ranges from 0 to 4,294,967,295.

Table 9.2: Data types for integers

Q. What is floating point data? Describe different data types for storing floating point data.

The floating point data is a numeric value with decimal point or fraction. It is also called real type data. It includes both positive and negative values. The minus sign - is used to indicate negative value. If no sign is used, the value is positive by default.

Examples

Some examples of real values are 10.5, 5.3 and -10.91 etc.

Types of Floating Point

C provides different types of floating point data. These are as follows:

1. float
2. double
3. long double

1. float Data Type

float data type is used to store real values. It takes four bytes in memory. Its range is from $3.4x10^{-38}$ to $3.4x10^{+38}$. It provides the precision of six decimal places.

2. double Data Type

double data type is used to store large real values. It takes eight bytes in memory. Its range is from $1.7x10^{-308}$ to $1.7x10^{+308}$. It provides the precision of fifteen decimal places.

3. long double Data Type

long double data type is used to store very large real values. It takes ten bytes in memory. Its range is from $1.7x10^{-4932}$ to $1.7x10^{+4932}$. It provides the precision of nineteen decimal places.

Data Type	Size in Bytes	Description
float	4	$3.4x10^{-38}$ to $3.4x10^{+38}$
double	8	$1.7x10^{-308}$ to $1.7x10^{+308}$
long double	10	$1.7x10^{-4932}$ to $1.7x10^{+4932}$

Table 9.3: Data types for float

Q. Describe the concept of overflow and underflow with examples.

Each type of numeric varible can store a particular minimum and maximum value. An error occurs if the value assigned to the variable is less than the minumum value or more than the maximum value.

An overflow occurs when the value assigned to a variable is more than the maximum possible value. An underflow occurs when the value assigned to a variable is less than possible the minimum value.

For example, an integer variable can store values from -32768 to 32767. If the assigned value is more than 32767, it is known as integer overflow. But if the assigned value is less than -32768, it is called integer underflow.

The overflow and underflow is not detected by the compiler. It may generate wrong results. It is the responsibility of the user to ensure that the values assigned to the variables are within the allowed range.

Program 9.1

Write a program that explains the concept of overflow and underflow.

```
#include <conio.h>
#include <stdio.h>
void main()
{
    short testVar = 32767;
    clrscr();
    printf("%d \n", testVar);
    testVar = testVar + 1;
    printf("%d \n", testVar);
    testVar = testVar - 1;
    printf("%d \n", testVar);
    getch();
}
```

Output:

```
32767
-32768
32767
```

Q. Describe different problems that may occur while working with floating point numbers.

The user may face some problems while working with floating point numbers. These problems are as follows:

1. Cancellation Error

The cancellation error occurs due to the manipulation of very large and very small floating numbers. The manipulation may show unexpected result. The larger number may cancel out the smaller number when both numbers are added. Suppose the user adds the numbers 1970.0 and 0.0000001243. The result of this addition may be 1970.000000 on some computers. In this result, the value 0.0000001243 has been cancelled in the calculation.

2. Arithmetic Underflow

The arithmetic underflow occurs when arithmetic calculation is performed on two very small numbers. The result may be too small to be represented in a particular variable. The result may be represented as zero in this situation.

3. Arithmetic Overflow

The arithmetic overflow occurs when calculation is performed on two very large numbers. The result may be too large to be represented in a particular variable. A garbage value may appear in this situation.

Q. Explain scientific or exponential notation for floating point numbers.

The floating point numbers can also be written in exponential or scientific notation. This notation represents large floating point numbers in a short way.

Exponential Notation

The exponential notation *consists* of two parts:

- Mantissa
- Exponent

The general form of writing floating point values in exponential notation is as follows:

±m **e** ±n OR ±m **E** ±n

The value before **e** is known as **mantissa** and the value after **e** is known as **exponent.**

Some important points about exponential notation are as follows:

- Mantissa and exponent may be positive or negative value.
- Exponent is always an integer value. It cannot be a real number.
- The absolute value of mantissa must be greater or equal to 1 and less than 10.

Suppose we have a floating point number 1000.0. It can be represented in scientific notation as 1.0×10^3. It can also be written in exponential form as **1.0e3.** In this example, 1.0 is mantissa and 3 is the value of exponent. Similarly, $0.004694 = 4.694 \times 10^{-3}$. It can also be written as **4.694e-3.**

Some examples of floating point numbers with scientific and exponential notations are as follows:

Floating Point Number	Scientific Notation	Exponential Notation
1.98	1.98×10^0	1.98e0
4,679,000.0	4.679×10^6	4.679e6
0.00053	5.3×10^{-4}	5.3e-4

Q. Explain range and precision.

The possible values that a floating type variable can store are described in terms of precision and range:

- **Precision:** Precision is the number of digits after the decimal point.
- **Range:** Range is the exponential power of 10.

A float variable has a precision of 6 digits and a range of 10^{-38} to 10^{+38}. Suppose we have a value 0.1234567×10^3. The precision of this value is 7 and range is 3. Similarly, a value 0.123456×10^{-3} has a precision of 6 digits and a range of -3.

A **double** variable has a precision of 15 digits and a range of 10^{-308} to 10^{+308}. A large precision provides more accuracy. The **double** data type is more suitable than **float** if the value to be stored requires more precision.

Q. Describe character data type of C language.

Character Data Type

char data type is used to store character value. It takes 1 byte in memory. It is used to represent a letter, number or punctuation mark and a few other symbols.

Character values are normally given in single quotes. It can represent individual characters such as 'a', 'x', '5', and '#'. The character '5' is manipulated differently than integer 5. It is possible to perform mathematical operation on character values. The characters can be added, subtracted and compared like numbers.

Example

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

```
void main()
{
  char ch1, ch2, sum;
  clrscr();
  ch1 = '2';
  ch2 = '6';
  sum = ch1 + ch2;
  printf("Sum = %d", sum);
  getch();
}
```

Output:
Sum = 104

The ASCII values of '2' and '6' are 50 and 54 respectively. That is why the result of the above example is 104. The printable characters in ASCII are from 32 to 126. The value 32 is the code of blank space. The value 126 is the code of ~ symbol. The other codes represent non-printable control characters.

The characters are stored in ASCII code form. **ASCII** stands for **American Standard Code for Information Interchange**. The ASCII code values of characters are used when they are added, subtracted or compared.

The keywords **singed** and **unsigned** can be used with **char** data type. The singed characters can represent numbers from -128 to 127. The unsigned characters can represent number from 0 to 255. English alphabets, numbers and punctuation marks are always represented with positive numbers.

Q. What are operators? List different types of operators in C.

Operators are the symbols that are used to perform certain operations on data. C provides a variety of operators. Different types of operators in C are as follows:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Increment and Decrement Operators
- Compound Assignment Operators

Q. What is arithmetic operator? Explain different arithmetic operators in C.

Arithmetic Operator

Arithmetic operator is a symbol that performs mathematical operation on data. C language provides many arithmetic operators. Following is a list of all arithmetic operators used in C language:

Operation	Symbol	Description
Addition	+	Adds two values
Subtraction	-	Subtracts one value from another value
Multiplication	*	Multiplies two values
Division	/	Divides one value by another value
Modulus	%	Gives the remainder of division of two integers

Table 9.4: Mathematical operators in C language

Use of Arithmetic Operators

All arithmetic operators work with numeric values. Suppose we have two variables A and B where A = 10 and B = 5. The arithmetic operators can be applied on these variables as follows:

Operation	Result
A + B	15
A – B	5
A * B	50
A / B	2
A % B	0

Table 9.5: Use of mathematical operators

Some important points about modulus operator are as follows:

- Modulus operator is also called remainder operator.
- The modulus operator works only with integer values.
- If modulus operator is used with the division of 0, the result will always be 0. For example the expression 0 % 5 will give 0 as a result.
- In expression like 3 % 5, 3 is not divisible by 5. Its result is 3.

Q. Briefly explain the working of division operator.

The manipulation of division operator is different from other operators. The result of division operation is always an integer if both the divisor and dividend are integers. The fractional part of the quotient is truncated.

Example

The result of 7.0/2.0 is 3.5 but the result of 7/2 is 3. The floating point number must be used to get the accurate result of a division operation.

Q. What is expression? Also explain arithmetic expression.

Expression

A statement that evaluates to a value is called an expression. An expression gives a single value. An expression consists of operators and operands. An operator is a symbol that performs some operation. Operand is the value on which the operator performs some operation. Operand can be a constant or variable etc. An expression may consist of any number of operators and operands.

Arithmetic Expression

A type of expression in which only arithmetic operators are used is called arithmetic expression. An arithmetic expression may contain integers and floating point numbers.

Examples

Some examples of expressions are as follows:

```
A + B;
```

division
m / n;
x + 100;

In the expression A + B, + is the operator. A and B are variables used as operands in the expression.

Q. What is meant by data type of an expression?

Data Type of Expression

The data type of an expression depends on the types of operands in the expression. For example, the result of an expression is int if both operands are integers.

An expression in which operands are of different data types is called mixed-type expression. In this case, the result of an expression is evaluated to larger data type in the expression. Suppose an expression contains an integer and double as operands. The result will be evaluated to double data type.

Some examples of data type of expressions are as follows:

Expression	Data Type of Expression
int + float	float
int – long	long
int * double	double
float / long double	long double

Q. Describe assignment statement.

A statement that assigns a value to a variable is known as assignment statement. The assignment operator = is used in assignment statement to assign a value or computational result to a variable.

The name of variable is written on the left side of the assignment operator and the value is written on the right side of assignment operator. The value can be a constant, variable, expression or function.

Syntax

The syntax of writing an assignment statement is as follows:

```
variable = expression;
```

variable	It is the name of variable to which the value is assigned.
=	It is the assignment operator used to assign the value to the variable.
expression	It is the expression whose returned value is assigned to the variable. It can be a constant, variable or combination of operands and arithmetical operators. The expression on right side is evaluated first. The result is then assigned to the variable on left side.

Examples

Some examples of assignment statements are as follows:

```
A = 100;
C = A + B;
X = C – D + 10;
```

Q. Describe Lvalue and Rvalue.

An lvalue is an operand that can be written on the left side of assignment operator =. It must always be a single value. An rvalue is an operand that can be written on the right side of assignment operator =. All lvalues can be used as rvalues. But all rvalues cannot be used as lvalues. For example, a constant can be used as rvalue but it cannot be used as lvalue. The expression x = 5 is valid but the expression 5 = x is not valid.

Program 9.2

Write a program that performs all mathematical operations on two variables.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a, b;
  a = 10;
  b = 5;
  clrscr();
  printf("a + b = %d \n",a + b);
  printf("a – b = %d \n",a – b);
  printf("a * b = %d \n",a * b);
  printf("a / b = %d \n",a / b);
  printf("a %% b = %d \n",a % b);
  getch();
}
```

Output:

```
a + b = 15
a – b = 5
a * b = 50
a / b = 2
a % b = 0
```

Q. Differentiate between unary and binary operators.

The difference between unary and binary operators is as follows:

Unary Operators

A type of operator that works with one operand is known as unary operator. Following operators are unary operators:

–, ++, --

The above operators are used with one operand as follows:

```
– a;
N++;
--x;
```

Binary Operators

A type of operator that works with two operands is known as binary operator. Following operators are binary operators:

+, –, *, /, %

The above operators are used with two operands as follows:

```
a + b;
x / y;
```

Q. Describe compound assignment statement.

An assignment statement that assigns a value to many variables is known as compound assignment statement. The assignment operator = is used in this statement.

Example

The following statement

```
A = B = 10;
```

assigns the value 10 to both A and B. Some examples of compound assignment statement are as follows:

```
x = y = z = 100;
m = n = 50;
```

Q. Describe compound assignment operators.

C language provides **compound assignment operators** that combine assignment operator with arithmetic operators. Compound assignment operators are used to perform mathematical operations more easily.

Syntax

The general syntax of compound assignment operator is as follows:

```
variable op = expression;
```

variable	The variable to assign a value.
op	It can be any arithmetic operator.
expression	It can be a constant, variable or arithmetic expression

For example, the statement:

```
N += 10;
```

is equivalent to

```
N = N + 10;
```

Both statements are equivalent and perform the same operation. Both statements add 10 in the current value of N. All other mathematical operators can also be combined with assignment operator.

Examples

Some examples of compound assignment are as follows:

A += 10	is equivalent to A = A + 10
A -= 10	is equivalent to A = A - 10
A *= 10	is equivalent to A = A * 10
A /= 10	is equivalent to A = A / 10
A %= 10	is equivalent to A = A % 10

Program 9.3

Write a program that performs all compound assignment operations on an integer variable.

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

```
void main()
{
  int a;
  clrscr();
  a = 10;
  printf("Value of a: %d \n", a);
  a += 5;
  printf("Value of a after a+=5: %d \n", a);
  a -= 5;
  printf("Value of a after a-=5: %d \n", a);
  a *= 2;
  printf("Value of a after a*=2: %d \n", a);
  a /= 2;
  printf("Value of a after a/=2: %d \n", a);
  a %= 2;
  printf("Value of a after a%%=2: %d \n", a);
  getch();
}
```

Output:

```
Value of a : 10
Value of a after a+=5 : 15
Value of a after a-=5 : 10
Value of a after a*=2 : 20
Value of a after a/=2 : 10
Value of a after a%=2 : 0
```

Q. Describe increment operator.

The increment operator is used to increase the value of a variable by 1. It is denoted by the symbol ++. It is a unary operator and works with single variable.

The increment operator cannot increment the value of constants and expressions. For example, A++ and X++ are valid statements but 10++ is an invalid statement. Similarly, (a+b)++ or ++(a+b) are also invalid. Increment operator can be used in two forms:

- Prefix Form
- Postfix Form

Prefix Form

In prefix form, the increment operator is written before the variable as follows:

```
++y;
```

The above line increments the value of variable y by 1.

Postfix Form

In postfix form, the increment operator is written after the variable as follows:

```
y++;
```

The above line also increments the value of variable y by 1.

Difference between Prefix & Postfix Increment

When increment operator is used independently, prefix and postfix form work similarly. For example, the result of A++ and ++A is same. But when increment operator is used in a larger expression with other operators, prefix and postfix forms work differently. For example, the results of two statements *A = ++B* and *A = B++* are different.

The statement A = ++ B has two operators ++ and =. It works in the following order:

1. It increments the value of B by 1.
2. It assigns the value of B to A.

The above statement is equivalent to the following two statements:

```
++B;
A = B;
```

In postfix form, the statement **A = B++** works in the following order:

1. It assigns the value of **B** to **A**.
2. It increments the value of **B** by 1.

The above statement is equivalent to the following two statements:

```
A = B;
B++;
```

Program 9.4

Write a program that explains the difference of postfix increment operator and prefix increment operator used as independent expression.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a, b, x, y;
  clrscr();
  a = b = x = y = 0;
  a++;
  b = a;
  ++x;
  y = x;
  printf("a = %d \n b = %d \n", a, b);
  printf("x = %d \n y = %d \n", x, y);
  getch();
}
```

Output:

```
a = 1
b = 1
x = 1
y = 1
```

Program 9.5

Write a program that explains the difference of postfix increment operator and prefix increment operator used as part of a larger expression.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a, b, x, y;
  clrscr();
  a = b = x = y = 0;
  b = a++;
  y = ++x;
  printf("a = %d \n b = %d \n", a, b);
  printf("x = %d \n y = %d \n", x, y);
  getch();
}
```

Output:

```
a = 1
b = 0
x = 1
y = 1
```

Q. Describe decrement operator.

The decrement operator is used to decrement the value of a variable by 1. It is denoted by the symbol - -. It is a unary operator and works with single variable.

The decrement operator cannot decrement the value of constants and expressions. For example, A- - and X- - are valid statements but 10- - is an invalid statement. Similarly, (a+b)- - or - -(a+b) are also invalid. Decrement operator can be used in two forms:

1. Prefix Form

In prefix form, decrement operator is written **before** the variable as follows:

```
--y;
```

The above line decrements the value of variable y by 1.

2. Postfix Form

In postfix form, the decrement operator is written after the variable as follows:

```
y--;
```

The above line also decrements the value of variable y by 1.

Difference between Prefix & Postfix Decrement

When decrement operator is used independently, prefix and postfix form work similarly. For example, the result of A- - and - -A is same. But when increment operator is used in a larger expression with other operators, prefix and postfix forms work differently. For example, the results of two statements A = - -B and A = B- - are different. The statement A = - - B contains two operators - - and =. It works as follows:

1. It decrements the value of B by 1.
2. It assigns the value of B to A.

The above statement is equivalent to the following two statements:

```
--B;
A = B;
```

In postfix form, the statement A = B- - works in the following order:

1. It assigns the value of B to A.
2. It decrements the value of B by 1.

The above statement is equivalent to the following two statements:

```
A = B;
B--;
```

Program 9.6

Write a program that explains the difference of postfix decrement operator and prefix decrement operator used as independent expression.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a, b, x, y;
  clrscr();
```

```
    a = b = x = y = 0;
    a--;
    b = a;
    --x;
    y = x;
    printf("a = %d \n b = %d \n", a, b);
    printf("x = %d \n y = %d \n", x, y);
    getch();
}
```

Output:
a = - 1
b = -1
x = -1
y = -1

Program 9.7

Write a program that explains the difference of postfix decrement operator and prefix decrement operator used as part of a larger expression.

```
#include <stdio.h>
#include <conio.h>
void main() → main function.
{
  int a, b, x, y;
  clrscr(); clear screen
  a = b = x = y = 0;
  b = a--;
  y = --x;
  printf("a = %d \n b = %d \n", a, b);
  printf("x = %d \n y = %d \n", x, y);
  getch(); answer
}
```

Output:
a = -1
b = 0
x = -1
y = -1

Q. Define relational operators. How many relational operators are provided in C language? Also describe relational expression.

The relational operators are used to specify conditions in programs. A relational operator compares two values. It produces result as true or false. The relational operators are sometimes called comparison operators as they test conditions that are either true or false.

C provides the following six basic relational operators:

Operator	Description
>	Greater than operator returns true if the value on left side of > is greater than the value on the right side. Otherwise returns false.
<	Less than operator returns true if the value on left side of < is less than the value on right side. Otherwise returns false.
==	Equal to operator returns true if the values on both sides of = are equal. Otherwise returns false.
>=	Greater than or equal to operator returns true if value on left side of >= is greater than or equal to the value on right side. Otherwise returns false.
<=	Less than or equal to operator returns true if the value on left side of <= is less than or equal to the value on right side. Otherwise returns false.
!=	The not equal to operator. Returns true if the value on the left side of <> is not equal to the value on the right. Otherwise returns false.

Table 9.6: Relational Operators

Relational Expression

A relational expression is a statement that uses relational operators to compare two values. The result of a relational expression can be true or false. Both sides of a relational expression can be a constant, variable or arithmetic expression.

Examples

Suppose the value of A = 10 and the value of B = 5. Different relational expressions will be evaluated according to the given table:

Relation Expression	Result
A > B	True
A < B	False
A <= B	False
A >= B	True
A == B	False
A != B	True

Table 9.7: Examples of relational expressions

Q. Explain compound condition with the help of example.

A type of comparison in which more than one conditions are evaluated is called compound condition. It is used to execute a statement or set of statements by testing many conditions.

Example

For example, a program inputs two numbers. It displays OK if one numbers is greater than 100 and second number is less than 100. Compound condition is executed by using logical operators.

Q. What are logical operators? Discuss different logical operators in C language.

The logical operators are used to evaluate compound conditions. There are three logical operators in C language:

1. AND operator (&&)
2. OR operator (||)
3. NOT operator (!)

1. AND Operator (&&)

The symbol used for AND operator is (&&). It is used to evaluate two conditions. It produces true result if both conditions are true. It produces false result if any one condition is false.

Condition 1	Operator	Condition 2	Result
False	&&	False	False
False	&&	True	False
True	&&	False	False
True	&&	True	True

Example

Suppose we have two variables A = 100 and B = 50. The compound condition (A>10) && (B>10) is true. It contains two conditions and both are true. So the whole compound condition is also true.

The compound condition (A>50) && (B>50) is false. It contains two conditions. One condition (A>50) is true and second condition (B>50) is false. So the whole compound condition is false.

2. OR Operator (| |)

The symbol used for OR operator is (| |). It is used to evaluate two conditions. It gives true result if either condition is true. It gives false result if both conditions are false.

Condition 1	Operator	Condition 2	Result
False	||	False	False
False	||	True	True
True	||	False	True
True	||	True	True

Example

Suppose we have two variables A = 100 and B = 50. The compound condition (A>50) || (B>50) is true. It contains two conditions and one condition (A>50) is true. So the whole compound condition is also true. The compound condition (A>500) || (B>500) is false because both conditions are false.

3. NOT Operator (!)

The symbol used for NOT operator is (!). It is used to reverse the result of a condition. It gives true result if the condition is false. It gives false result if the condition is true.

Operator	Condition	Result
!	True	False
!	False	True

Example

Suppose we have two variables A = 100 and B = 50. The condition !(A==B) is true. The result of (A==B) is false but NOT operator converts it into true. The condition !(A>B) is false. The condition (A>B) is true but NOT operator converts it into false.

Q. Explain operator precedence with example.

The order in which different types of operators in an expression are evaluated is known as operator precedence. It is also known as hierarchy of operators.

Each operator has its own precedence level. If an expression contains different types of operators, the operators with higher precedence are evaluated before the operators with lower precedence. The order of precedence in C language is as follows:

- Any expression given in parentheses is evaluated first.
- Then multiplication * and division / operators are evaluated.
- Then plus + and minus - operators are evaluated.

- In case of parentheses within parentheses, the expression of the inner parentheses will be evaluated first.

Operator	Precedence
! (Logical Not)	Highest
*, /, %	
+, -	
<, <=, >, >=	
==, !=	
&&	
||	
= (Assignment Operator)	Lowest

Table 9.8: Operator Precedence

The above table shows that logical NOT operator has the highest priority. It is a unary operator. The assignment operator has the lowest priority.

Example

The expression 10 * (24 / (5 – 2)) + 13 is evaluated in the following order:

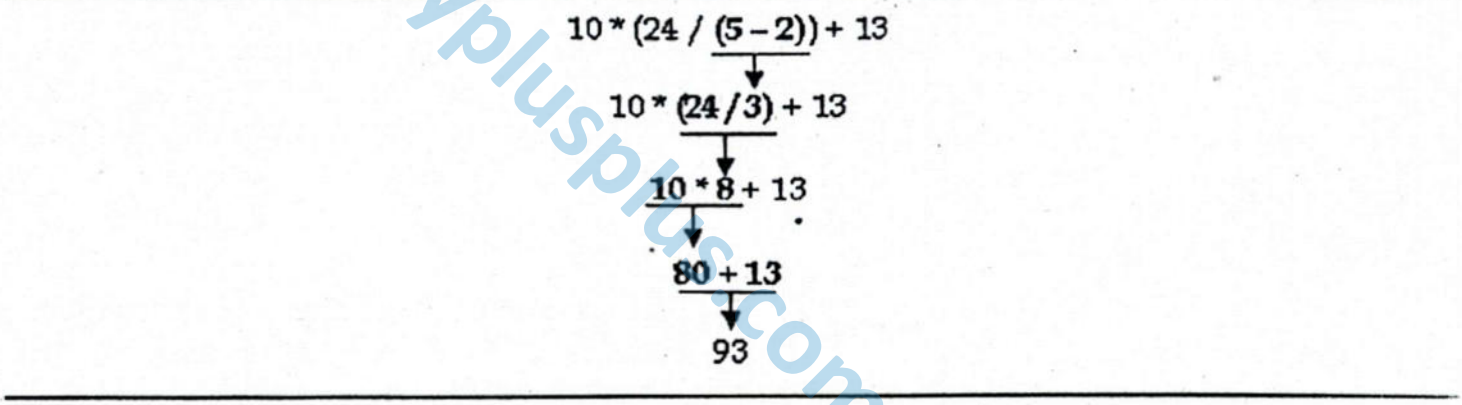

Figure 9.3: Operator precedence

1. First of all, the expression 5–2 will be evaluated. It gives a value 3.
2. Secondly, 24 will be divided by the result of last line i.e. 24 / 3 giving value 8.
3. Thirdly, 10 will be multiplied by 8 i.e. giving a result 80.
4. Finally, 80 will be added in 13 and the last result will be 93.

Q. Explain associativity of operators.

The order in which operators of same precedence are evaluated is known as operator associativity. If an expression contains some operators that have same precedence level, the expression is evaluated either from left-to-right or right-to-left. Operator associativity in C language is as follows:

Operators	Associativity
() ++ (postfix) - - (postfix)	Left-to-right
+ (unary) - (unary) ++ (prefix) - - (prefix)	Left-to-right
* / %	Left-to-right
+ -	Left-to-right
= += -= *= /=	Right-to-left

Table 9.9: Associativity of operators

Example

The expression 10 * 24 / 5 – 2 + 13 contains four operators. Two operators * and / have equal precedence. These expressions will be evaluated from left to right. Similarly, two operators + and - also have equal precedence.

The expression will be evaluated in the following order:

- First of all, the expression 10 * 24 is evaluated. It gives a value 240.
- Secondly, 240 will be divided by 5 i.e. 240 / 5 giving a value 48.
- Thirdly, 2 will be subtracted from 48 i.e. 48 – 2 giving a result 46.
- Finally, 46 will be added in 13 and the last result will be 59.

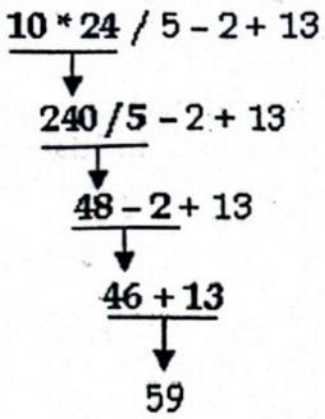

Figure 9.4: Operator associativity

Q. Describe comments and their use in C programs.

Comments are the lines of program that are not executed. The compiler ignores comments and does not include them in the executable program. That is why the comments do not affect the size of executable program.

Comments are used to increase the readability of the program. Comments are notes about different lines of code that explain the purpose of the code. The user can insert information notes in the code. It helps in debugging and modifying the program later.

Comments can be added anywhere in the code. Comments can be added in programs in two ways:

- **Sing-line Comments:** Comments on single line are added by using double slash "//". Any thing written on the right side of double slash is considered as comments and is ignored during execution.
- **Multi-line Comments:** Multi-line comments are inserted to the code by placing /* at the beginning of the comments. The character */ is used to end the multi-line comments.

Examples

Following line is an example of comments.

```
// Practice makes a man programmer.
```

Another way to add comments on single line is as follows:

```
/* These are also comments... */
```

In order to add comments on many lines in program, following way is used:

```
/* These lines are
for the purpose of
writing comments */
```

Q. What is type casting?

The process of converting the data type of a value during execution is known as type casting. Type casting can be performed in two ways:

1. Implicit type casting
2. Explicit type casting

1. Implicit Type Casting

Implicit type casting is performed automatically by the C compiler. The operands in arithmetic operation must be of similar types. If the data types of operands are different, the value with lower data type is converted into higher data type.

Different types of variables are as follows:

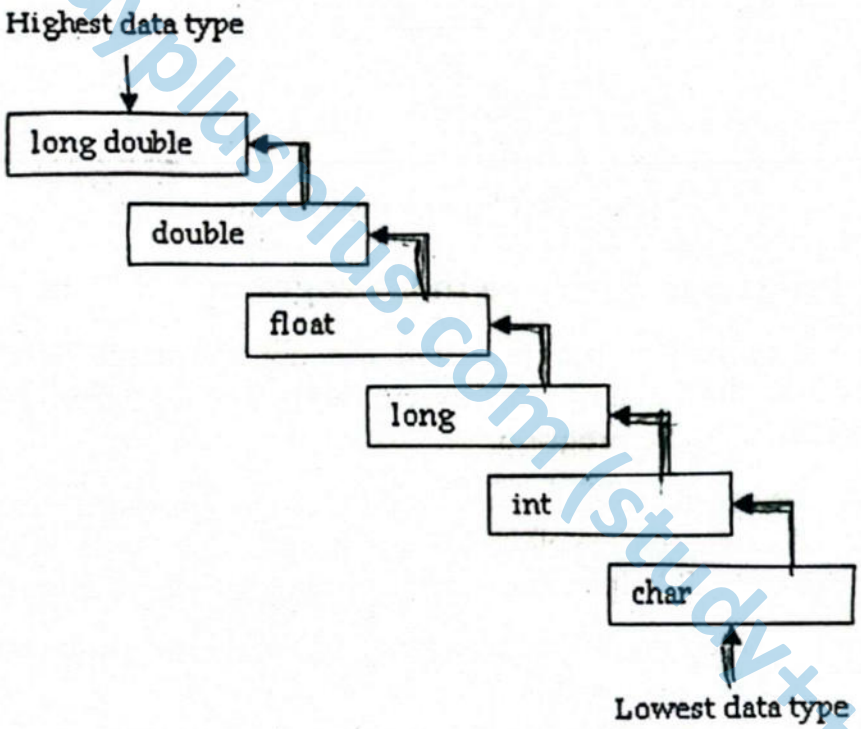

Figure 9.5: Type casting order

Example

Suppose x is an integer variable and y is a long variable and the following expression is evaluated:

x + y

In the above expression, data type of x is lower than the data type of y. So the value of x will be converted into long during the evaluation of expression. The data type of x is not changed. Only the data type of the value of x is changed during the evaluation of the expression.

2. Explicit Casting

Explicit casting is performed by programmer. It is performed by using **cast operator.** The cast operator tells the computer to convert the data type of a value.

Syntax

The syntax of using cast operator is as follows:

```
(type) expression;
```

type	It indicates the data type to which operand is to be converted.
expression	It indicates the constant, variable or expression whose data type is to be converted.

Example

Suppose **x** and **y** are two **float variables. x** contains 10.3 and **y** contains 5.2 and the following expression is evaluated:

```
x % y
```

The above expression will generate an error because data type of **x** and **y** is **float.** The modulus operator cannot work with float variables. It only works with integers. So the expression can be written as follows:

```
(int) x % (int) y
```

The above statement converts the values of **x** and **y** into integers and evaluates the expression. The value of **x** will be converted to 10 and value of **y** will be converted to 5. So the result of the above expression will be 0.

Program 9.8

Write a program that divides two float variables and finds the remainder by using explicit casting.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  float a, b;
  int c;
  clrscr();
  a = 10.3;
  b = 5.2;
  c = (int)a % (int)b;
  printf("Result is %d", c);
  getch();
}
```

Output:
Result is 0

How above Program Works?

The above program finds the remainder of two **float** variables using modulus operator. The modulus operator cannot work with float variables. So type casting is used *to* convert these variables into integers in the expression. The result of the expression will be 0.

Q.10. What data type would you use to represent the following items?

- Number of children at your school
- A letter grade on an exam
- The average marks of your class

The data types used to represent the above items are as follows:

Item	Data Type
Number of children at your school	int
A letter grade on an exam	char
The average marks of your class	float

Q.11. Which of the following are valid variable names in C?

- income — Valid.
- total marks — Invalid. A variable cannot have spaces in it.
- double — Invalid. A keyword cannot be used as variable name.
- averge-score — Invalid. A variable cannot have hyphen in it.
- room# — Invalid. A variable cannot have hash sign in it.
- _area — Valid.
- no_of_students — Valid.
- long — Invalid. A keyword cannot be used as variable name.
- Item — Valid.
- MAX_SPEED — Valid

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

Invalid expression	Valid Expression
z = 4.0 w * y;	z = 4.0 * w * y;
y = yz	y = y * z;
a = 6b4;	a = 6 * b * 4;
c = 3(a+b);	c = 3 * (a + b);
z = 7w + xy;	z = 7.0 * w + x * y;

Q.14. Assume that you have the following variable declarations:

```
int a, b, c, d, p;
float v, w, x, y, z;
```

Evaluate each of the following satements assuming that:

a is 2, z is 1.3, c is 1, d is 3 and y is 0.3E+1.

Expression	Result
v = a * 2.5 / y	1.666667
w = a / y;	0.666667
p = a / d;	0
x = (a + c) / (z + 0.3);	-3.000000
b = d / a + d % a;	2
y = c / d * a;	0.000000

Short Questions

Q.1. What is an identifier? Give some examples of identifier.

The identifiers are the names used to represent variable, constants, types, functions and labels in the program. An identifier in C may consist of any number of characters. But the first 31 characters are significant to C compiler. Some examples of identifiers are Student_Age, Item20, Sum etc

Q.2. Write the legal characters for an identifier.

The legal characters to construct identifier are as follows:

a. The upper alphabets from A to Z.
b. The lower alphabets from a to z.
c. The digits from 0 to 9.
d. The underscore character

Q.3. List different types of identifiers in C.

There are two types of identifiers in C language. These are Standard Identifiers and User-defined Identifiers.

Q.4. Define standard identifier and give some examples.

A type of identifier that has special meaning in C is known as **standard identifier**. C cannot use a standard identifier for its original purpose if it is redefined. printf and scanf are examples of standard identifiers.

Q.5. Define user-defined identifier and give some examples.

The type of identifier that is defined by the programmer to access memory location is known as user-defined identifier. The user-defined identifiers are used to store data and program results. Some examples of user-defined identifiers are num, age, marks etc.

Q.6. Define keyword and give some examples.

Keyword is a word in C language that has a predefined meaning and purpose. The meaning and purpose of a keyword is defined by the developer of the language. It cannot be changed or redefined by the user. Examples include if, while, int, and const.

Q.7. Define variable. Why is it used in programs?

A variable is a named memory location or memory cell. It is used to store program's input data and its computational results during execution. The value of variable may change during the execution of the program. However, the name of variable cannot be changed.

Q.8. Define variable declaration. Is it compulsory to declare all variables?

The process of specifying the variable name and its type is called **variable declaration**. A variable must always be declared before it can be used in a program. The compiler gives an error if an undeclared variable is used in a program.

Q.9. Why is C known as strongly typed language?

C is a strongly typed language. It means that a variable must always be declared before it can be used in a program. The compiler gives an error if an undeclared variable is used in a program.

Q.10. What is specified in variable declaration?

- **Variable Name:** It refers to the memory location of the variable.
- **Variable Data Type:** It indicates the type of data that can be stored in variable.

Q.11. Differentiate between declaring and defining a variable.

The variable declaration specifies the name and type of the variable to the compiler. It does not set aside memory location. The variable definition specifies the name and type of the variable as well as sets aside memory location to store the data in the variable.

Q.12. Define variable initialization? How is a variable initialized?

The process of assigning a value to a variable at the time of declaration is known as variable initialization. The equal sign = is used to initialize a variable. Variable name is written on left side and the value is written on the right side of equal sign.

Q.13. Define constant.

A constant is a quantity that cannot be changed during program execution.

Q.14. Distinguish between a constant and a variable.

The value stored in a variable can be changed during the execution of the program. The values stored in constants cannot change.

Q.15. List different types of constants available in C language. Give examples.

C language provides two types of constants. These are numeric constants and character constants. The examples of numeric constants are 10, -5 and 5.7 etc. The examples of character constants are 'A', '9', '=' and '$' etc.

Q.16. What are integer constants? Give some examples.

Integer constants are numeric values without fraction or decimal point. Both positive and negative integer constant are used. The minus sign - is used for negative integer constants. Some examples of integer constants are 87, 45, -10 and -5.

Q.17. What are floating point constants?

Floating point constants are numeric values with fraction or decimal point. Floating point constants represent values that are measured. Both positive and negative floating point constants are used. The minus sign – is used for negative floating point constants.

Q.18. Define character constants. Give some examples.

Any character written in single quotation mark is called character constant. All alphabetic characters, digits and special symbols can be used as character constants. Some examples of character constants are 'A','9','=' and '$' etc.

Q.19. Define string constants. Give some examples.

A collection of characters is called string. String constant is written in double quotes. It may consist of any alphabetic characters, digits and special symbols. Some examples of sting constants are "Pakistan", "123" and "99-Mall Road, Lahore".

Q.20. What do you known about the data type in C language?

Data type defines a set of values and a set of operations on these values. The computer manipulates various types of data. The data and its type are defined before designing the actual program that is used to process the data. The type of each data value is identified at the beginning of program design.

Q.21. Name and give the purpose of basic data type available in C.

int are used to store whole numbers (integers). float is used to store a real or floating point value (value with a decimal point). A char stores a single character including letters, punctuation marks and digits.

Q.22. List at least three data types in C.

The three data types in C language include char, int and float.

Q.23. How much memory does different data types require (Any three)?

The char data type requires 1 byte. The int data type requires 2 bytes. The float data type requires 4 bytes.

Q.24. Why is it important to assign a data type to a variable?

Variables are assigned data types to specify the amount of memory occupied by that variable. Different data types occupy different amount of memory. Using proper data type for a variable can save the memory.

Q.25. List out different rules for declaring variables in C.

The first character of variable must be a letter or underscore. Blank spaces are not allowed in variable names. Variable may include letters, numbers and underscore (_). Reserved word cannot be used as variable name.

Q.26. Give syntax of declaration of any character type variable.

The syntax of initializing a variable is as follows:

```
char variable;
```

char — It is the keyword to declare character type variable.

variable — It is name of the variable to be declared.

Q.27. Give some examples of valid variable names.

income	_area	no_of_students
MAX_SPEED	Integer	Marks1

Q.28. Give some examples of invalid variable names.

total marks	double	room#
My.school	2ndTry	$cost

Q.29. Which are two ways to use data types in C?

A data type that is predefined in the language is called **standard data type**. Some examples of standard data type are **int, float, long** and **char** etc.

Q.30. Define user-defined data type.

C also allows the user to define his own data types known as **user-defined data type.**

Q.31. Which data types are used for storing integer data?

The data types used for storing integer data are int, short int, long int, unsigned int and unsigned long int.

Q.32. Which data types are used for storing floating point data?

The data types used for storing floating point data are float, double and long double.

Q.33. Which problems may occur while working with floating point numbers?

These problems are cancellation error, arithmetic underflow and arithmetic overflow.

Q.34. Why does cancellation error occur?

The cancellation error occurs due to the manipulation of very large and very small floating numbers are manipulated. The manipulation may show unexpected result. The larger number may cancel out the smaller number when both numbers are added.

Q.35. How does an arithmetic underflow occur?

Arithmetic underflow occurs due to the manipulation of two very small numbers. The result may be too small to be represented when two very small numbers are manipulated. The result is represented as zero in this situation. For example, an integer variable can store values from -32768 to 32767. An underflow will occur if the assigned value is less than -32768.

Q.36. What is the cause of an arithmetic overflow?

The arithmetic overflow occurs due to the manipulation of two very large numbers. The result may be too large to be represented when two very large numbers are manipulated. A garbage value may appear in this situation. For example, an integer variable can store values from -32768 to 32767. An overflow will occur if the assigned value is more than 32767

Q.37. State the use of character data type.

char data type is used to store character value. It takes 1 byte in memory. It is used to represent a letter, number or punctuation mark and a few other symbols.

Q.38. How are characters stored?

The characters are stored in ASCII code form. ASCII stands for American Standard Code for Information Interchange. The ASCII code values of characters are used when they are added, subtracted or compared.

Q.39. What is the use of operators?

Operators are the symbols that are used to perform certain operations on data. C provides a variety of operators. These include arithmetic operators, relational operators, logical operators, bitwise operators etc.

Q.40. List different types of operators in C.

Different types of operators in C are arithmetic operators, relational operators, logical operators, assignment operators, increment and decrement operators and compound assignment operators.

Q.41. Define arithmetic operator and list different arithmetical operators in C.

Arithmetic operator is a symbol that performs mathematical operation on data. C language provides many arithmetic operators. These are + , -, *, / and %.

Q.42. What is arithmetic expression?

A type of expression in which only arithmetic operators are used is called arithmetic expression. An arithmetic expression may contain integers and floating point numbers.

Q.43. What is the use of assignment operator?

The assignment operator is used to store a value or computational result in a variable. The symbol = is used to represent the assignment operator.

Q.44. What does the symbol = do in C?

The symbol = is called assignment operator. It is used to assign values to variables in C.

Q.45. Differentiate between z='a' and z=a?

The expression z='a' will assign the character 'a' to variable z. The expression z=a will assign the value of variable a to variable z.

Q.46. Differentiate between s="word" and s=word?

The expression s="word" will assign the string "word" to variable s. The expression s=word will assign the value of variable word to variable s.

Q.47. What is the use of assignment statement?

A statement that assigns a value to a variable is known as assignment statement. The assignment operator = is used in assignment statement to assign a value or computational result to a variable.

Q.48. Define compound assignment statement.

An assignment statement that assigns a value to many variables is known as compound assignment statement. The assignment operator = is used in this statement.

Q.49. What is the use of increment operator?

The increment operator is used to increase the value of a variable by 1. It is denoted by the symbol ++. It is a unary operator and works with single variable.

Q.50. Define prefix Increment operator.

The prefix increment operator is used to decrement the value of a variable by 1. It is a unary operator and works with single variable. In prefix form, the increment operator is written before the variable like ++y;

Q.51. What is the use of decrement operator?

The decrement operator is used to decrement the value of a variable by 1. It is denoted by the symbol - -. It is a unary operator and works with single variable.

Q.52. Differentiate between preincrement and postincrement operators ++x and x++.

Preincrement operator increments the value of x then uses the new value of x in the statement. Postincrement operator uses the current value of x in the statement and then performs the increment.

Q.53. Differentiate between unary and binary operators.

The unary operators work with one operand such as ++x. The binary operators work with two operands such as x + y.

Q.54. Define relational operator? List different relational operators in C language.

The relational operators are used to specify conditions in programs. They compare two values and produce result as true or false. C provides six relational operators. These operators are >, <, ==, >=, <=, and !=.

Q.55. What is relational expression? Give some examples.

Relational expression is a statement that uses relational operators to compare two values. Examples of relational expressions are A>B, A<B, A<=B, A>=B, A==B and A!=B.

Q.56. What is compound condition?

A type of comparison in which more than one conditions are evaluated is called compound condition. It executes a statement or set of statements by testing many conditions.

Q.57. Define logical operators.

Logical operators are used to evaluate compound conditions. The logical operators in C language are AND operator (&&), OR operator (| |) and NOT operator (!).

Q.58. What is the use of AND operator?

The symbol used for AND operator is (&&). It is used to evaluate two conditions. It produces true if both conditions are true. It produces false result if any one condition is false.

Q.59. What is the use of OR operator?

The symbol used for OR operator is (| |). It is used to evaluate two conditions. It produces true if either condition is true. It produces false result if both conditions are false.

Q.60. What is the use of NOT operator?

The symbol used for NOT operator is (!). It is used to reverse the result of a condition. It produces true result if the condition is false. It produces false result if the condition is true.

Q.61. What is meant by operator precedence?

The order in which different types of operators in an expression are evaluated is known as operator precedence. It is also known as hierarchy of operators. Each operator has its own precedence level.

Q.62. What is associativity of operators?

The order in which operators of same precedence are evaluated is known as operator associativity. If an expression contains some operators that have same precedence level, the expression is evaluated either from left-to-right or right-to-left.

Q.63. What is expression?

A statement that evaluates to a value is called an expression. An expression gives a single value. An expression consists of operators and operands. An operator is a symbol that performs an operation. An expression may consist of any number of operators and operands.

Q.64. What is the data type of an expression?

The data type of the result of an expression is called data type of an expression. It depends on the types of operands in the expression. The result of an expression is int if both operands are integers.

Q.65. Define mixed-type expression.

An expression in which operands are of different data types is called mixed-type expression. In this case, result of an expression is evaluated to larger data type in expression.

Q.66. Which lines of C program are not executed?

Comments are the lines of program that are not executed. The compiler ignores comments and does not include them in the executable program. That is why the comments do not affect the size of executable program.

Q.67. Why are comments used in C program?

Comments are used to increase the readability of the program. Comments are notes about different lines of code that explain the purpose of the code. The user can insert information notes in the code. It helps in debugging and modifying the program later.

Q.68. Define type casting and its types.

The process of converting the data type of a value during execution is known as type casting. The types casting are Implicit type casting and Explicit type casting.

Q.69. Define implicit type casting.

Implicit type casting is performed automatically by the C compiler. The operands in arithmetic operation must be of similar types. If the data types of operands are different, the value with lower data type is converted into higher data type.

Q.70. Define explicit casting.

Explicit casting is performed by programmer. It is performed by using cast operator. The cast operator tells the computer to convert the data type of a value.

Q.71. Distinguish between implicit and explicit type casting.

Implicit type casting is performed automatically by the C compiler. If the data types of operands are different, the value with lower data type is converted into higher data type. Explicit casting is performed by programmer. It is performed by using cast operator.

Q.72. Where comments can be added in programs?

Comments can be added anywhere in the code. Comments can be added in programs in two ways. The user can write comments on single line or on multiple lines.

Q.73. How are comments added on single line?

Comments on single line are added by using double slash "//". Any thing written on the right side of double slash is considered as comments and is ignored during execution.

Q.74. How are comments added on multiple lines?

Comments on multiple lines are added by using "/*" and "*/" symbols. Any thing written between two symbols is considered as comments and is ignored during execution.

Q.75. Why comments do not affect the size of program?

The compiler ignores comments and does not include them in the executable program. That is why the comments do not affect the size of executable program.

Q.76. Write a statement to declare an integer variable i initialized to 10.

Ans: int i = 10;

Q.77. Write a C constant declaration that gives the name PI to the value 3.142.

Ans: const float PI = 3.142;

Q.78. Differentiate between "area" and area in C language.

The literal string "area" is a string constant value. The identifier area is a variable to store a value.

Q.79. Write a declaration statement and an assignment (input) statement to assign 30 meter to a variable named length.

int length;
length = 30; OR int length = 30;

Q.80. Declare two integer variables in one declaration statement. The two integer variables should be named testNumber and testScore.

Ans: int testNumber, testScore;

Q.81. Write the following expression in C language: $a = 2bc^3$

Ans: a = 2 * b * c * c * c

Q.82. Write the following expression in C language:

$$x = \frac{3y}{5-z}$$

Ans: x = (3 * y)/(5 – z) OR x = 3 * y / (5 – z)

Q.83. Find out of the expression 10*(24/(5-2))+13:

Ans: 93

Q.84. Write the mathematical expression in C expression. Ke=1/2mv²:

Ans: Ke = 1 / 2 * m * (v * v);

Q.85. Write the following statement in C: $z = area^{\sqrt{area}}$

Ans: z = area *sqrt(area);

Q.86. What is the value of y after the following code executes:

float y = 3.4 + sqrt (25.0);

Ans: 8.4

Q.87. Rewrite the following expression without using compound assignment operator +=:

x += 2;

Ans: x = x + 2;

Q.88. Rewrite the following expression without using the operator ++:

another++;

Ans: another = another + 1;

Q.89. Rewrite the following expression using a compound assignment operator:

result = result / 2;

Ans: result /= 2;

Q.90. If a and b are integers and b = 12, which value is given to a by following expression?

```
a = 7.8 + b / 5;
```

Ans: 9

Q.91. Determine the value of variable num after the following statements execute.

```
int num = 10;
num += 5;
num - = 2;
num *= 3;
printf("%d", num);
```

Ans: 39

Q.92. Predict the output of the following code:

a.

```
int number = 6;
number++;
printf(%d\n",number)
```

Ans: 7

b.

```
int number = 6;
++number;
printf("%d\n",number);
```

Ans: 7

c.

```
int number = 6;
int x = 0;
x = number- -;
printf("%d\n",x)
```

Ans: 6

d.

```
int number = 6
int x = 0;
x = --number;
printf("%d\n",x);
```

Ans: 5

Q.93. Predict the output of the following code:

a.

```
int x=20,y=35;
x = y++ + x++;
y = ++y + ++x;
printf("%d%d",x,y);
```

Ans: 5794

b.

```
int x=5;
printf("%d,%d,%d",x,x*4,x%4);
```

Ans: 5,20,1

c.

```
int x=10, y=15;
x = x++;
y = ++y;
printf("%d %d",x,y);
```

Ans: 11,16

d.

```
int  i=12;
int  j=46;
int  k=56;
printf("i=%d ",i);
printf("j=%d ",j);
printf("k=%d ",k);
```

Ans: i=12 j=46 k=56

Q.94. Trace the output of the following C programs:

a.

```
main() {
  int a,b;
  a=77;
  b=40;
  a+=b;
  printf("a=%d",a); }
```

Ans: a = 117

b.

```
int i, j, z;
i=10;
j=3;
z=i%j;
printf("%d",z);
```

Ans: 1

Q.95. Suppose a=15 and a+=3 is executed. What is new value of a?

Ans: 18

Q.96. What are the values of x and y after execution of the following statements?

```
x = 5;
y = ++x;
```

Ans: x = 6, y = 6

Q.97. What will the value of x be after the following statements execute?

```
int x = 0;
int y = 5;
int z = 4;
x = y + z * 2;
```

Ans: 13

Q.98. Write the following statement in C code: p and q are less than zero

Ans: `(p < 0) && (q < 0)`

Q.99. Write the following statement in C code: score is between 50 and 60 inclusive

Ans: `(score >= 50) && (score <= 60)`

Q.100. What will be the value of z after executing the following statements?

```
int x;
z = 18.0 / 4;
```

Ans: 4.5

Q.101. Write the following statements as single statement?

```
int x = 5;
int y = 10;
int z = 15;
```

Ans: `int x = 7, y = 16, z = 28;`

Q.102. What is the final value of i in the following code?

```
int i = 40;
i /= (i / 4 + 70);
```

Ans: 0

Q.103. Write the output of the following statement:

a. (7%2) * (++x)	where x = 2	**Ans: 3**
b. (n<10) || (n>25)	where n = 25	**Ans: 0 (False)**
c. a || b && c	where a=5, b=10, c=15	**Ans: 1 (True)**

Q.104. Write the output of the following statement:

a. (x<y) && (z=5)	when x=10, y=15, z=5	**Ans: 1 (True)**
b. (x>y) || (z=5)	when x=0, y=15, z=5	**Ans: 1 (True)**

Q.105. Find the value of:

a. (i>0) && (j<5)	where i=8, j=5	**Ans: 0 (False)**
b. (x+y) / x-y	where x=2, y=4	**Ans: -1**
c. a*3/(b*a)	where a=4, b=2	**Ans: 1**

Q.106. What is the value of x be after the following statements execute?

```
int x = 0;
int y = 5;
int z = 10;
x = y + z + 5;
```

Ans: 20

Q.107. What is the output of the following?

```
int n, a, b;
n = 500;
a = n % 100;
b = n / 10;
n = n % 10;
printf("%d %d %d", n, b, a);
```

Ans: 0500

Q.108. What is the output of the following?

```
int x = 25;
int y = 4;
printf("%d", x / y);
printf("%d", y / x);
printf("%d", x % y);
```

Ans: 601

Q.109. What is the output of the following?

```
int i = 5;
printf("%d %d %d", i, i++, ++i);
```

Ans: 5 5 7

Multiple Choices

1. **A memory location with some data that can be changed is called:**
   a. Constant. (b) Variable.
   c. Named constant. d. None
2. **Variables are created in:**
   (a). RAM b. ROM. c. Hard disk. d. None.
3. **A memory location with some data that cannot be changed is called:**
   (a). Constant. b. Variable. c. Keyword d. None.
4. **Which of the following is a valid character constant?**
   a. a. b. "Hello" (c) '6' d.=
5. **Which of the following operators has lowest precedence?**
   a. ! b. + (c). = d. = =
6. **C is strongly typed language, it means that:**
   a. Every program must be compiled before execution
   (b) Every variable must be declared before it is being used
   c. The variable declaration also defines the variable
   d. Sufficient data types are available to manipulate each type of data
7. **a + =b is equivalent to:**
   a. b+=a b. a=+b (c) a=a+b d. b=b+a
8. **Which is true about a variable?**
   a. The name and data value can both change.
   b. The name can change, but the data value cannot.
   (c) The name cannot change, but the data value can.
   d. The name and the data value both cannot change.

9. **Which is NOT a rule for naming variables?**
   a. Use a descriptive name for the variable.
   b. Start the name of a variable with a letter.
   c. Use nothing but letters, digits, or the underscore character.
   d. All of the above

10. **Variable and constant names can not contain a(n):**
   a. Number b. Underscore c. Letter d. Period

11. **Variable names can not begin with a(n):**
   a. Number b. Underscore
   c. Upper-case letter d. Lower-case letter

12. **Which of the following is NOT a valid identifier?**
   a. return b. myInt c. myInteger d. total3

13. **Which of the following are valid variable names?**
   a. long b. Integer
   c. notlongenough d. Both b and c

14. **Which term describes the kind of values that a variable can store?**
   a. Variable Name b. Datatype
   c. Variabletype d. Variablesize

15. **Which statement is true about data types?**
   a. Each data type has no memory requirements.
   b. Each data type has different memory requirements.
   c. Each data type has the same memory requirements.
   d. None of the above

16. **Which data type is used to store numeric value with no decimal point?**
   a. int b. char c. float d. All

17. **Which is a numeric data type?**
   a. Floating point b. Integer
   c. Both a and b. d. None

18. **The number of bytes used by int data type in C is?**
   a. 2 b. 8 c. 12 d. 16

19. **The number of bytes used by long int data type in C is?**
   a. 2 b. 4 c. 12 d. 16

20. **An integer variable can store the value:**
   a. -1.1 b. "123" c. 32898 d. None

21. **The integer, long and short data types are known as:**
   a. Integer data types b. Non-integral data types
   c. float data types d. Non-numeric data types

22. **The data type that can handle decimal places is:**
   a. long b. float c. char d. None

23. **The float, long float and double data types are known as:**
   a. Integer data b. Character data
   c. Integral data d. Real data

24. **The number of bytes used by float data type in C is:**
   a. 2 b. 4 c. 12 d. 16

25. **The number of bytes used by double data type in C is:**
   a. 2 b. 8 c. 12 d. 16

26. What happens when the result of a calculation exceeds the capacity of data type?
a. System error b. Logic error c. Syntax error d. Overflow

27. The exponential notation consists of:
a. Mantissa b. Exponent c. Range d. a and b

28. The number of digits after a decimal point is called:
a. Significance b. Precision c. Range d. Scope

29. Which of the following data type is used to store string value?
a. char b. float c. string d. None

30. The number of bytes used by char data type in C is:
a. 2 b. 1 c. 12 d. 16

31. Which variable should be used to store the value "I want an A in this exam"
a. char b. int c. float d. character

32. Another way to write the value 3452211903 is:
a. 3.452211903e09 b. 3.452211903e-09
c. 3.452211903x09 d. 3452211903e09

33. The constant 0.15e+6 represents the same value as:
a. 150000.0 b. 6.15 c. 0.75 d. 0.21

34. Which of the following statements is NOT legal?
a. char ch='b'; b. char ch='0';
c. char ch=65; d. char ch="cc";

35. Which of the following are required to declare a variable?
a. keyword b. Variable name
c. Data type d. Both b & c

36. Which of the following is used to separate each variable while declaring more than one variable on the same line?
a. Commas b. Colons
c. Pipes d. Semicolons

37. Which is a valid statement for declaring a variable?
a. int marks; b. int a,b,c;
c. double salary; d. All

38. Which of the following data types is most appropriate for storing a name?
a. float b. Integer c. char d. None

39. A process of assigning initial value to a variable at the time of declaration is called:
a. Assigning b. Initializing c. Naming d. None

40. Which is a valid statement for initialization of a variable?
a. int n=100; b. int x=50, y=75;
c. char grade='a'; d. All

41. Which is NOT a valid statement to initialize a variable?
a. int =100; b. long population=15000;
c. char n[]="Hello word"; d. const int N=100;

42. Which of the following statements is correct?
a. float num1; num2; b. int day, night;
c. int continue = 5.0; d. string black = 'white';

43. Which of the following constant definitions is correct?
a. const float PRICE = 3,500; b. const int LENGTH = 6.5;
c. const float COST = $700.0; d. const int MY HEIGHT = 60 / 12;

44. Which of the following can be used as character constant:
a. Characters b. Digits
c. Special characters d. All

45. Which of the following are valid examples of character constant?
a. 'A' b. '9' c. '$' d. All

46. A character constant must be enclosed in:
a. Quotation marks (").
b. Single quotes (').
c. Exclamation points (!).
d. Pound signs (#).

47. What is the valid range of numbers for int type of data?
a. 0 to 256
b. -32768 to +32767
c. 65536 to +65536
d. No specific range

48. How many bytes are occupied by the statement: float a, b;
a. 1 byte b. 4 c. 8 d. 16

49. An expression consists of:
a. Operators b. Operand c. Both a and b d. None

50. An expression consists of:
a. Constant b. Variable
c. Arithmetic operators d. All

51. All of the following are valid expressions EXCEPT:
a. Sales – Profit b. Mile, Gallons
c. Pi * Radius d. M/n

52. An assignment statement will:
a. Perform a calculation.
b. Store the results of a calculation.
c. Display the results of a calculation.
d. Both a and b.

53. Which of the following is NOT arithmetic operator?
a. + b. - c. % d. >

54. The modulus operator is used for:
a. Exponentiation b. Multiplication
c. Division d. Integer remainder

55. Which of the following operators works only with integer value?
a. % b. + c. - d. /

56. The expression 10%3 has a value equal to ;
a. 1 b. 3 c. 8 d. None

57. The expression 3%5 has a value equal to:
a. 3 b. 5 c. 0 d. None

58. The expression 0%4 has a value equal to:
a. 0 b. 4 c. Both a and b d. None

59. The value of the C expression 9 / 5 * 2 is:
a. 3.6 b. 0.9 c. 0 d. 2

60. The value of the C expression 5 / 9 * 2 is:
a. 0.27 b. 1.11 c. 0 d. None

61. The value of the C expression 13 + 21 % 4 - 2 is:
a. 0 b. 12 c. 16 d. 14

62. What is the value of the following expression? 20.0 * (9/5) + 32.0
(a) 52.0 b. 54.0 c. 51.0 d. 50

63. Which statement can be used to store letter A in char variable someChar?
a. someChar = "A"; (b) someChar = 'A';
c. someChar = A; d. a and c

64. Which of the following operators is used to assign a value to a variable?
a. > b. + (c) = d. *

65. Which of the following is correct syntax for writing an assignment statement?
(a) Variable = right_side b. Constant = right_side
c. Constant = variable d. None

66. The left side of an assignment statement holds:
(a) Variable. b. Constant.
c. Expression. d. Both a and b.

67. The right side of an assignment statement holds:
a. A variable (b) An expression
c. Both a and b d. None

68. Which of the following is valid assignment statement?
a. a = 100; b. c = a + b;
c. x = c – d + 10; (d) All of these

69. Which of the following operators works with one operand?
(a) Unary b. Binary c. Ternary d. None

70. Which of the following operators works with two operands?
a. Unary (b) Binary c. Ternary d. None

71. Which of the following statements assigns a value to many variables?
(a) Compound assignment b. Lvalue
c. Rvalue d. Input statement.

72. The statement I += 3 has the same effect as:
(a) I = I+3. b. I = 3. c. I-3 = I. d. I3.

73. To add a value 1 to variable y, you write:
a. y += 1; b. y = y + 1; c. y = 1 + y; (d) All

74. To add number to sum, you write:
a. sum += number; b. sum = sum + number;
c. sum = Number + sum; (d) Both a and b

75. Which is a correct translation of the following algebraic expression into a C?

$$\frac{1}{c} + \frac{1-a}{1+b}$$

a. 1 / c + 1 - a / 1 + b b. 1 / c + (1 - a / 1 + b)
c. (1 / c) + (1 - a / 1 + b) (d) 1 / c + (1 - a) / (1 + b)

76. Which of the following operator is used to increase the value of a variable?
(a) ++ b. +- c. >> d. None

77. Which of the following operator is used to decrease the value of a variable?
(a) - - b. -+- c. << d. None

78. Which choice has the highest order of precedence when computing an expression?
(a) () b. * c. / d. +

79. If x is int and y is float, what promotion will occur in the expression: y = 5.0 + x;
a. float to int
b. int to float
c. float to double
d. No promotion

80. What is the name for a word that has a specific meaning in C?
a. Keywords
b. Comments
c. Token
d. Operators

81. Another name for keywords is:
a. Reserved words
b. Special words
c. Comments
d. None

82. A relation expression produces results as:
a. True
b. False
c. Either a or b
d. Any value

83. Which of the following relation operator represents Greater-then-or-Equal-to?
a. >=
b. =>
c. >==
d. <==

84. Which of the following relation operator represents Not equal to?
a. !=
b. =!
c. #==
d. <=

85. The number of relational operators in C language is:
a. 4
b. 6
c. 3
d. 1

86. Which of the following is a relational operator?
a. !=
b. ||
c. &&
d. +

87. All of the following are relational operators EXCEPT:
a. !=.
b. ||.
c. =
d. >=.

88. Which of the following returns True if A = 25 and B = 35:
a. A+B
b. A>=B
c. A!=B
A=B

89. For A = 4 and B = 4 which evaluates as true?
a. A != B
b. A < B
c. A > B
d. A >= B

90. What operator is used in a compound condition to join two conditions?
a. Relational Operator
b. Logical Operator
c. Relational Results
d. Logical Results

91. Operators that perform operations on Boolean data are:
a. Arithmetic operators
b. Logical operators
c. Relational operators
d. All

92. Which of the following symbol is used for OR operator?
a. ||
b. &&
c. !
d. None

93. Which of the following symbol is used for AND operator?
a. ||
b. &&
c. !
d. None

94. Which of the following symbol is used for NOT operator?
a. ||
b. &&
c. !
d. None

95. All of the following are logical operators EXCEPT:
a. NOT
b. AND
c. OR
d. =

96. For A = 3 and B = 7 which evaluates to true?
a. !(A < B)
b. !(B < 6)
c. (A = 3) && (B = 5)
d. (A = B) || (A > B)

97. Which of the following is equivalent to (p >= q)?
a. p < q
b. !(p < q)
c. p > q
d. !p < q

98. The total number of keywords in C is:
a. 32 b. 60 c. 90 d. None

99. Logical NOT operator, denoted by !, is a:
a. Ternary operator b. Bitwise operator
c. Binary operator d. Unary operator

100. Logical operators are:
a. NOT b. AND c. OR d. All

101. In C language, the symbol = represents:
a. Comparison operator b. Assignment operator
c. Equal to operator d. None

102. Which of the following data types offers the highest precision?
a. float b. long int
c. long double d. unsigned long int

103. When the result of the computation of two very small numbers is too small to be represented, it is called:
a. Arithmetic overflow b. Arithmetic underflow
c. Truncation d. Round off

104. Which of the following is a valid C statement?
a. char ch = 'a'; b. char ch = 'ab';
c. car ch = 97; d. char ch = "a";

105. In C language, two conditions can be joined by using:
a. Arithmetic operators b. Relational operators
c. Logical operators d. Bitwise operators

106. Data type int contains bytes:
a. 10 b. 4 c. 2 d. 16

107. These are operators that add and subtract one from their operands.
a. ++ and - - b. Plus and minus
c. Binary and unary d. None

108. Which of the following is an illegal variable name?
a._customer_num b. jan2009
c. dayOfWeek d. 2dGraph

109. Which of the following is used to declare variables that can hold real numbers:
a. int b. float c. real d. None

110. The numeric data types in C can be divided into two general categories:
a. Integer and floating point b. Single and double
c. Numbers and characters d. None

111. Which of the following is called the modulus operator?
a. * b. % c. + d. &

112. Which operator is used to represent equality in C language?
a. = b. == c. >< d. None

113. The characters or symbols that perform operations on operands are called:
a. Operators b. Syntax c. Operation code d. None

114. The = operator in C language indicates:
a. Assignment b. Subtraction c. equality d. Negation

115. Associativity is either right to left or:
a. Left to right b. Top to bottom c. Back to front d. None

116. When a variable is assigned a number that is too large for its data type, it:
a. Overflows
b. Exceeds expectations
c. Underflows
d. None

117. const, double and int and examples of:
a. Comments
b. Reserved Words
c. Compiler Directives
d. None

118. Which of the following operator has the highest priority?
a. !=
b. !
c. &&
d. ||

119. Which of the following has the highest precedence?
a. ++
b. &&
c. ||
d. -

120. Precedence determines which operator:
a. is most important
b. is used first
c. is faster
d. operates on largest number

121. The equivalent statement of sum=sum-num is:
a. sum= -num
b. sum=num
c. sum –=num
d. sum=- -num

122. Which address operator indicates the address of a variable?
a. %
b. \a
c. #
d. &

123. Comments are used to increase the _____of the program?
a. Visibility
b. Beauty
c. Readability
d. Complexity

Answers

1. b	2. a	3. a	4. c	5. c	6. b
7. c	8. c	9. d	10. d	11. a	12. a
13. D	14. B	15. B	16. a	17. c	18. a
19. b	20. d	21. a	22. b	23. d	24. b
25. b	26. d	27. d	28. b	29. a	30. b
31. a	32. a	33. a	34. d	35. d	36. a
37. d	38. c	39. b	40. d	41. a	42. b
43. d	44. d	45. d	46. b	47. b	48. c
49. c	50. d	51. b	52. b	53. d	54. d
55. a	56. a	57. a	58. a	59. d	60. c
61. b	62. a	63. b	64. c	65. a	66. a
67. b	68. d	69. a	70. b	71. a	72. a
73. d	74. d	75. d	76. a	77. a	78. a
79. b	80. a	81. a	82. c	83. a	84. a
85. b	86. a	87. c	88. c	89. d	90. b
91. b	92. a	93. b	94. c	95. d	96. b
97. b	98. a	99. d	100. d	101. b	102. c
103. b	104. a	105. c	106. c	107. a	108. d
109. b	110. a	111. b	112. b	113. a	114. a
115. a	116. a	117. b	118. b	119. a	120. b
121. c	122. d	123. c			

Fill in the Blanks

1. _____ are the names used to represent variables, constants, types, functions and label s in C programs.
2. Total keywords in the C language are _____.
3. Keywords are always written in _____.
4. C is a _____ language because all variables must be declared before being used.
5. First character of a variable must be a _____in C language.
6. The maximum length of a character constant is _____.
7. The _____ defines a set of values and set of operations on those values.
8. The int type data takes_____ bytes in the memory.
9. The int can store integer values from_____ to_____.
10. The unsigned int can store integer value from_____ to _____.
11. The float type data takes_____ bytes in memory.
12. Modulus operator is also called_____ operator.

Answers

1. Identifier	2. 32	3. Lower case	4. Strongly type
5. letter	6. 1 character	7. Data type	8. Two
9. -32768,32767	10. 0,65635	11. four	12. remainder

True/ False

1. printf and scanf are standard identifiers.
2. In C language, all variables must be declared before being used.
3. Standard data types are not predefined in C language.
4. The double data type takes 8 bytes in memory.
5. The symbol for modulus operator is %.
6. The symbol = is used to compare two values.
7. Operator precedence determines order of evaluation of operators in an expression.
8. For many compilers a C variable name can be up to 31 characters.
9. Modulus operator cannot work with float values.
10. Result of the expression 3%10 is 10.
11. '+' & '++' are two binary operators.
12. The operator that works with three operands is called ternary operator.
13. Conditional operator is a binary operator.
14. C program can only use lowercase letters in variable name.
15. Standard data types are not predefined in C language.

Answers

1. T	2. T	3. F	4. T	5. T
6. F	7. T	8. T	9. T	10. F
11. F	12. T	13. F	14. T	15. F