**“Elements of C Language”**

**ASSIGNMENT**

**II**

**Prepared by:** Srijal Dangol **Submitted To:** Ashim Sir

**Roll Number:** 17 **Subject:** C Programming

**Shift:** Morning

**BscCSIT079**

1. **Write short notes Signed and Unsigned short Integers.**

* **Signed short Integers**

The term "signed" in computer code indicates that a variable can hold negative and positive values. By default all short integers are signed. Signed short Integers represents both positive and negative integers. The data type qualifier is signed short int or short int or short E.g. Variables are defined as signed short int a; short int b; short c; . It reserves 16- bits (2 bytes) in memory. This makes it useful for programs that need to conserve memory or work with large amounts of numerical data. The range of signed short integers is -2 15 To 215 -1,i.e -32768 to 32761, because of its limited range, signed short integers may not be appropriate for all applications, and programmers may need to use larger data types to represent larger numbers. The conversion character id d or i.

* **Unsigned short Integers**

An unsigned short integer is a data type in computer programming that represents a whole number between 0 and 65,535. The term "unsigned" means that the data type can only represent positive numbers. Like the signed short integer, the "short" designation means that the data type uses a small amount of memory to store the number, typically 2 bytes. This makes it useful for programs that need to conserve memory or work with large amounts of numerical data. However, because it can only represent positive numbers, the unsigned short integer may not be appropriate for all applications, and programmers may need to use larger data types to represent negative or larger numbers. The data type qualifier is unsigned short int or unsigned short. The conversion character is u.

1. **Write short notes Signed and Unsigned long Integers.**

* **Signed long Integer**

A signed long integer is a data type in computer programming that represents a whole number between - 2,147,483,648 and 2,147,483,647. The term "signed" means that the data type can represent both positive and negative numbers. The term "long" means that the data type uses a larger amount of memory to store the number which means it reserves 32-bits (4 byte). This makes it useful for programs that need to work with very large numerical data or require a wider range of values. However, because of its larger size, signed long integers may take up more memory and processing power, so programmers must consider the trade-offs between accuracy and efficiency when choosing a data type. The data type qualifier is signed long int or long int or long. E.g. variables are defined as – signed long int a; long int b; long c; The conversing character is Id.

* **Unsigned Long Integer**

An unsigned long integer is a data type in computer programming that represents a whole number between 0 and 4,294,967,295. The term "unsigned" means that the data type can only represent positive numbers. Like the signed long integer, the "long" designation means that the data type uses a larger amount of memory to store the number which means it reserves 32-bit(4 Byte). This makes it useful for programs that need to work with very large numerical data or require a wider range of values. However, because it can only represent positive numbers, the unsigned long integer may not be appropriate for all applications, and programmers may need to use larger data types to represent negative numbers. The data type qualifiers is unsigned long int or unsigned long. E.g. variables are defined as -unsigned long int a; Unsigned long b; . The conversion character is Iu.

1. **Write short notes Double Precision and Long Double Precision Floating point.**

* **Double Precision Floating Point**

When the accuracy provided by a float number is not sufficient, the type double can be used to define the number . A Double data type number uses 64 bits ( 8 byte) giving a precision of 14 digit. These are known as double precision number. Double represents fractional number of the range - 1.7e308 to +1.7e308. The data type qualifier is double . E.g. Variables are defined as -double a; . Its conversion character is If. Operations on double precision floating point numbers can be slower than operations on integers or single precision floating point numbers due to the larger amount of memory required. Double precision floating point numbers are commonly used in scientific and engineering applications where high precision is required.

* **Long Double Precision Floating point**

Long double precision floating point is a computer data type that is used to represent real numbers with an even higher degree of precision than double precision. Long double precision floating point numbers are typically represented using 80 bits or more of memory. The exact format of long double precision floating point numbers can vary between different compilers and operating systems. Long double precision floating point numbers can represent numbers with even more significant digits than double precision, sometimes up to 19 or 20 digits. Operations on long double precision floating point numbers can be slower than operations on double precision floating point numbers due to the larger amount of memory required. Long double precision floating point numbers are rarely used in everyday programming, but may be useful in specialized applications where extremely high precision is required.

1. **Write short notes Real Constants and also fractional form constants and exponential form constants mantissa/exponents).**

* **Real Constant**

In programming, a real constant is a literal value that represents a real number. Real constants can be expressed in various formats, such as decimal, scientific notation, or hexadecimal. Real constants are typically used to assign fixed values to variables or as operands in mathematical operations. Real constants are sometimes called floating point constants, as they are often stored in memory as floating point numbers. Depending on the programming language and the data type used, real constants may have a limited precision or range of representable values. Real constants can be combined with arithmetic operators, comparison operators, and other programming constructs to perform complex computations and make decisions based on numerical values.

* **Fractional Form Constant**

Fractional form constants are a type of real constant that are expressed as a fraction of two integer values. Fractional form constants are often used in programming languages to represent rational numbers, which are numbers that can be expressed as a ratio of two integers. In fractional form constants, the numerator and denominator of the fraction are typically separated by a forward slash (/) or other separator. Fractional form constants can be used to represent both proper fractions (where the numerator is smaller than the denominator) and improper fractions (where the numerator is larger than or equal to the denominator). When working with fractional form constants, it is important to handle cases where the denominator is zero or negative, as these can lead to division errors or undefined behavior. Fractional form constants can be converted to other data types, such as floating point numbers or integers, using appropriate conversion functions or operators.

* **Exponential Form Constant**

Exponential form constants are a type of real constant that are expressed in scientific notation, using a base and an exponent. Exponential form constants are often used in programming languages to represent very large or very small numbers, as they allow for compact and convenient notation. In exponential form constants, the base is typically a real number between 1 and 10, and the exponent is an integer. Exponential form constants can be written in various formats, such as "1.23E4" or "1.23e+4", depending on the programming language and conventions used. Exponential form constants can be positive or negative, depending on the sign of the exponent. Exponential form constants can be used in arithmetic operations, comparison operations, and other programming constructs, just like other types of real constants.

1. **Write short notes Symbolic Constants. Write rules for defining a symbolic constants. Also explain advantages of symbolic constants.**

* **Symbolic constant**

A Symbolic constant is a name that is used in place of a sequence of characters. The character may represent numeric constant , a character constant or a string constant . When a program is compiled, each occurrence of a symbolic constant is replaced by its corresponding character sequence . The symbolic constants are defined at the beginning of the program.

**Rules for defining a symbolic constant:**

* Symbolic constant names are same as variable names. Convention: Use capital letters while defining symbolic constants.
* No blank space permitted between # and name.
* A blank space is required between #define and symbolic name and between symbolic name and its value (i.e. constant)

**Some advantages of symbolic constant are:**

1. **Readability:** Symbolic constants can make code more readable and easier to understand, as they provide meaningful names for values that would otherwise be represented by hard-coded literals or magic numbers.
2. **Maintenance:** Symbolic constants can make code easier to maintain, as they provide a central location for values that may need to be changed in the future. Changing the value of a symbolic constant updates all references to that constant in the code, without the need to search and replace hard-coded literals.
3. **Consistency:** Symbolic constants can help enforce consistency in code, as they ensure that the same value is used consistently throughout the code. This can help avoid bugs and errors that may arise from using different values in different parts of the code.
4. **Safety:** Symbolic constants can improve the safety and correctness of code, as they can help prevent accidental modifications to values that should not be changed. Using symbolic constants instead of hard-coded literals can prevent unintended side effects and make code more predictable.
5. **Portability:** Symbolic constants can make code more portable, as they provide a way to abstract away platform-specific values or differences. By using symbolic constants, code can be written once and then easily adapted to different environments or platforms.