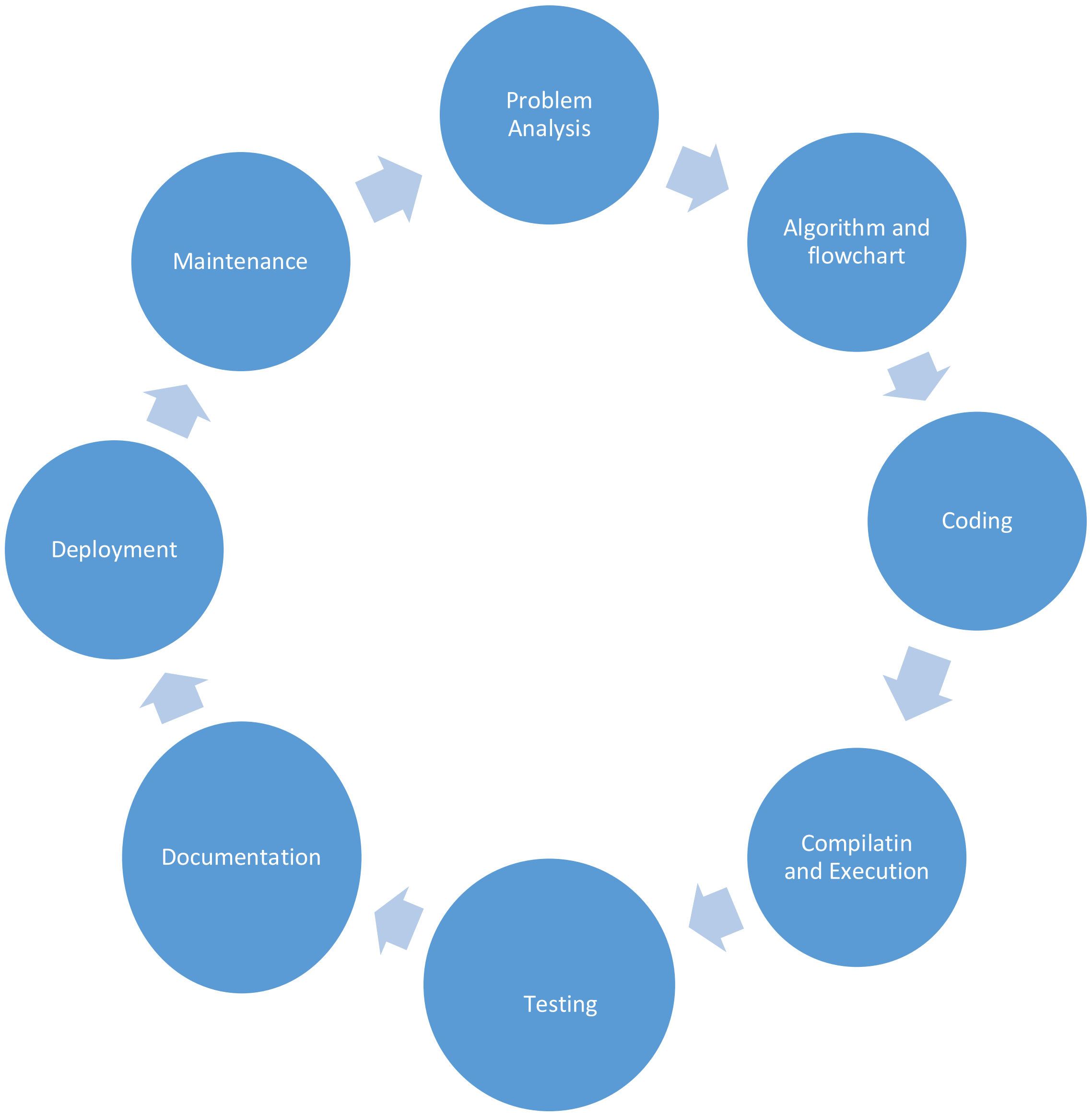
Software development Lifecycle

Software Development Life Cycle is a basically a detailed plan which describes how to create, maintain, alter and improve a specific software product (program).

SDLC consists of following activities:



1. **Problem Analysis:** It is important to give a clear and precise problem. In this step, we should specify the following things.
    - Objectives
    - Input requirements
    - Output requirements
    - Processing requirements
    - Evaluating feasibility
2. **Algorithm and flowchart:** Algorithm and flowchart helps to convert users' needs, logic or ideas into a suitable form, which helps the programmer in coding and implementation.
3. **Coding:** The coding is a process of transforming program logic or ideas into computer language format.
4. **Compilation and execution:**
   Compilation is the process of converting high level language into low level language because computer understand only low level language. Once, compilation is completed then the program is linked with another library files needed for execution.
5. **Debugging and Testing:** Debugging is the process of isolating and correcting any type of errors and testing ensures the program performs correctly the required task.
6. **Documentation:** Every step in the project is documented for future reference. Documentation of program helps to those who use, maintain and extend the program in future.
7. **Deployment:** The software is deployed after it has been approved for release.
8. **Maintenance:** Software Maintenance is the process of modifying a software for the following reason.
    - To remove the a bug from the program.
    - To improve the efficiency of code.
    - Add new function to fulfill new user requirements.

Data type modifier/Type Qualifiers

Data type modifiers or type Qualifiers are used to modify the properties of primitive or basic data types according to application requirements; so that we can be able to precisely utilize the computer memory.

Modifiers are prefixed with basic data types to modify the size of memory allocated for a variable. As the size varies the range of values stored by the variable also changes.

With the help of data type modifiers we can:

- **Modify the size**(i.e. the amount of memory to be allocated)
- **Modify the sign** (i.e decide only +ve or both +ve or –ve values can be stored)
- **Modify the range**(i.e .increase or decrease the range of values stored by data type)

- ✓ short and long are size modifiers
- ✓ signed and unsigned are sign modifiers.

short: short qualifier decreases the size of data type as well the range of values stored by the variable decreases.

long: Long qualifier increases the size of the basic data type as well the range values stored by variable increases

unsigned: Allows to store only +ve values.

signed: Allows to store +ve as well as –ve values. By default primitive data is prefixed with signed.

Primitive data type			Data type after using Modifiers		
Type	**Range**	**Size**	**Type**	**Range**	**size**
int	-32768 to +32767	2 bytes	signed int	-32768 to +32767	2 bytes
			unsigned int	0 to 65535	2 bytes
			long int	-2,147,483,648 to 2,147,483,647	4 bytes
			short int	-128 to +127	1 byte
			signed long int	-2,147,483,648 to 2,147,483,647	4 bytes
			unsigned long int	0 to 4,294,967,295	4 bytes
double	1.7E-308 to 1.7E+308	8 bytes	long double	3.4E -4932 to 1.1E +4932	10 bytes
char	-128 to +127	1 byte	signed char	-128 to +127	1 byte
			unsigned char	0 to 255	1 byte

How could you extend the range of the values represented by the basic data types?

OR

How can you modify the size of data type?

- Data type modifiers or type Qualifiers are used to modify the properties of primitive or basic data types.
  With the help of data type modifiers we can
  - **Modify the size** (i.e. the amount of memory to be allocated)
  - **Modify the range**(increase or decrease the range of values stored by data type)
  - **Modify the sign** (i.e decide only +ve or both +ve or –ve values can be stored)

  So, with the help of data type modifiers we can extend the range of values represented by the basic data types or modify the size of data type.

- There are four data type modifiers in C. They are

1. **long**
2. **short**
3. **unsigned**
4. **signed**

- ✓ Short and long are size modifiers
- ✓ Signed and unsigned are sign modifiers.

For example, storage space for **int** data type is 2 bytes with range -32768 to +32767 in 16 bit compiler. We can increase the range -2,147,483,648 to +2,147,483,647 by using **long int** which is of 4 bytes.

Similarly, we can decrease the range -128 to +127 by using **short int** which is 1 byte.

How can you declare the following variables using suitable data types? Mobile number, Distance jumped by frog, salary and Age. Also explain their type qualifier, memory occupancy size, conversion character and range.

Mobile number ***(Note: similar for registration number, symbol number, account number)***

As mobile number holds long digits of positive numbers so that **long int** data type will be suitable.

Type qualifier: mobile number holds only positive and large range of values .So that we can use the combination of unsigned and long i.e. **unsigned long** as a type qualifier.

Memory occupancy size: 4 bytes for 16 bit compiler

Conversion character(format specifier): %ld

Range of long int: -2,147,483,648 to +2,147,483,647

Distance jumped by frog: ***(Note: similar for weight, salary,body temperature)***

The distance jumped by frog is in decimal value so that we can we float as an datatype.

Type qualifier: No any type qualifier can be used with **float**

Memory occupancy size: 4 bytes for 16 bit compiler

Conversion character(format specifier): %f

Range of float: -$3.4*10^{38}$ to $3.4*10^{+38}$

Note: *for more precision of decimal numbers we can also use double as an datatype. We can use long as an type qualifier to increase the size as well range of values stored by double.*

Age:

As age is non-decimal value we use **int** as an datatype to store values.

Type qualifier: As age stores only positive values we can use **unsigned** as an type qualifier.

Memory occupancy size: 2 bytes for 16 bit compiler

Conversion character(format specifier): %d

Range of int: - 32768 to +32767

Mention the appropriate data type for storing the following data. Also justify your answer in brief.

i) A prime number between 5 and 555

As there exist many prime number between 5 and 555 the appropriate data type will be array of integers. Array being a collection of similar data elements and derived data type the prime number between 5 and 555 can be stored under a common variable name.

How can you declare following variables using suitable data types? Also explain each memory size and range

i) Address

The suitable data type for holding the address is string. As we know strings are sequence of characters stored in consecutive memory location. Each character in string occupies one byte of memory. Strings always terminated with null character '\0'.

So that the memory size required to holding address depends upon the number of characters in address.

Eg. The address named "Kathmandu" requires 10 bytes, including null character.

So, declaration,

char address[10];

is sufficient of hold address named Kathmandu.

As string being sequence of character, the range for each character in string is same as range of character i.e. -128 to +127