**CTags:** We can know were the function is declared.

\*To install CTags = $sudo apt install universal-ctags.

\*with grep command also we can find where the function is declared.

\*$ctags-R we get index file, load the index file data in vi.

\*In vi editor run the below command.

Set tags =<path -of-tag-file>

\*To navigate we use press ctrl-] and to navigate backwards use press control+t.

**Cscope**: It is similar to CTags.

\*To install scope : $sudo apt install scope.

\*cscope stores it data in file called cscope.out.

\*To come out of cscope press ctrl+d.

\*$find . -name “\*.[ch]” > cscope.files.

**Performance Profiling using gprof:** gprof allows us to measure how much percentage of time is spent in different functions. These way we can focus on optimizing the function.

\*$ gcc -o application -pg main.c.

\*We run the application as normal procedure

$ ./application

\*The run produces a file called gmon.out.

\*Generating performance report $gprof ./application gmon.out.

**Gdb:** The GNU debugger.

\*Debug is the runtime error.

\*While debugging we can use the break points.

**C Language:**

\*c language is discovered by Dennis Ritchie.

\*It is suitable for both system software and business packages.

\*Programs written in c are much faster and efficient.

**Structure of C:**

1)Documentation Section: When we create Document section it will be in comments. In Documents section we have

\*Description: Here we write what the program is all about.

\*Author

\*DOC/DOM

\*Version: New files start with 0.1v

2)Link section: Include of header files and macros.

3)Definition Section: Defining the prototyping of functions.

4)Global Declaration Section : Defining variables which can access globally.

5)Function Section : Here we declare functions.

**Constants, Variables, and datatypes:**The characters that can be used to form words, numbers and expressions depend upon the computer on which the program is run.

1)Characters (a-z)(A-Z)

2)Digits(0-9)

3)Special characters(@#$)

4)white spaces(like: blank spaces, carriage return, newline)

**CTokens:** The smallest individual units in a program are known as tokens.

They are: 1.keywords 2.Identifiers 3.constants 4.strings 5.special symbols 6.operators

**C Language is rich in its datatypes:**

**1)primary Datatypes:**

**Name Size(in bytes) Range of values**

Char 1 -128 to 127

Int 2 -32,768 to 32,767

Float 4 3.4e-38 to 3.4e+38

Double 8 1.7e-308 to 1.7e+308

**2)user-Defined Datatypes:**

\*Structures

\*unions

\*enum

**3)Derived Datatypes:**

\*Array

\*Function

\*Pointers

**Modifiers:** The basic datatypes may have several modifiers preceding them to serve the need of various situation. They are

a)signed

b)unsigned

c)long

d)short

may be applied to character and integer datatypes. However, the modifier long may also be applied to double.

After applying the modifiers to basic datatypes, their size as follows:

**Name size(in bytes) (range of values)**

Unsigned char 1 0 t0 255

Signed char 1 -128 to 127

Unsigned 2

Signed int 2

Short int 2

Long int 4

Long double 10

\*y = x++, first assign x value to y, then increment the x

\*y = ++x, first increment x, then assign.

**Ternary Operator:** op1?(op2):(op3)

Res=(exp1)?(exp2):(exp3);

Ex: int a=10

Int b=20

Int c=30

Res=(a>b)?((a>c)?a:c):((b>c)?b:c);

**Size of the Operator:**

The "size of the operator" typically refers to the sizeof operator in C. The sizeof operator is used to determine the size (in bytes) of a data type or an object in memory. It is a compile-time operator, meaning that the size is computed by the compiler, not during program execution.

sizeof(type)

sizeof expression

🡪sizeof(type) returns the size of the specified data type (e.g., int, char, float).

🡪 sizeof(expression) returns the size of the expression, which could be a variable or a more complex expression.