**2.1 Familiarization of Linux environment – How to do Programming in C with Linux**

Student can login into the system using the username and password then take the terminal, for getting command terminal, search for terminal or use shortcut key Ctrl+Alt+t. In terminal each student can login into the lab server using ssh command.

*ssh* login name@server IP address then press the enter key

Type password then press the enter key

After login ‘vi’ editor is used for doing programs.

*vi* program name.c

Press insert key to type the program

For saving the program, press Esc key then type *:wq*

For compile the program using cc command: *cc* program name.c

After successful compilation take the output of the program using *./a.out*

command.

LINUX COMMANDS

1. *ls* :- used to list the files and directories under the current directory.

Syntax:- *ls* press enter key

2. *rm*:- used to remove a particular file.

Syntax:- rm < file name>

3. *mkdir*:- used to create a directory.

Syntax:- mkdir <directory name>

4. *cd*:- used to change the directory.

Syntax:- cd <directory name>

5. *cd ..* :-used to leave from a particular directory.

Syntax:- cd .. press enter key

6. *rmdir*:- used to remove an empty directory.

Syntax:- rmdir <empty directory name>

**2.2 Introduction to C**

C is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs. The UNIX operating system, the C compiler, and essentially all UNIX applications programs have been written in C. The C has now become a widely used professional language for various reasons.

● Easy to learn

● Structured language

● It produces efficient programs

● It can handle low-level activities

● It can be compiled on a variety of computer platforms

Facts about C

C was invented to write an operating system called UNIX. C is a successor of B language which was introduced around 1970. The language was formalized in 1988 by the American National Standard Institute (ANSI). The UNIX OS was totally written in C by

1973. Today C is the most widely used and popular System Programming Language. Most of the state-of-the-art software have been implemented using C. Today's most popular Linux OS and RBDMS MySQL have been written in C.

Why to use C?

C was initially used for system development work, in particular the programs that make-up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

● Operating Systems

● Language Compilers

● Assemblers

● Text Editors

● Print Spoolers

● Network Drivers

● Modern Programs

● Databases

● Language Interpreters

● Utilities

The largest measure of C's success seems to be based on purely practical considerations:

1. The portability of the compiler

2. The standard library concept

3. A powerful and varied repertoire of operators

4. An elegant syntax

5. Ready access to the hardware when needed

6. And the ease with which applications can be optimized by hand-coding isolated procedures

Structure of a C program

Documentation Section

Link Section /Include header file section

Definition Section

Global declaration section

Main() function section

{

Declaration part

Executable part

}

Subprogram section

Function 1

Function 2

…

Function n

(User defined functions)

Documentation Section:-

Documentation section consists of a set of comment lines giving the name of the program, the author and other details.

Link section:-

Link section provides instructions to the compiler to link functions from the system library.

Definition Section:-

Definition section defines all symbolic constants. Global Declaration:-

This section declares some variables that are used in more than one function. This section also declares all the user defined functions.

Main() function section:-

Every C program must contain a main() function. This section contains two parts, declaration and executable parts. Declaration part declares all the variables used in the executable part. There is at least one statement in the executable part. Executable part contains the statements following the declaration of the variables.

Subprogram section:-

This section contains all the user defined functions that are called in the main() function. All sections, except the main () function and link section may be absent when they are not required.

Character set of C language

The character set in C Language can be grouped into the following categories.

1.Letters

2.Digits

3.Special characters

4.White Spaces

Programming in C Lab Manual **EST 102**

White Spaces are ignored by the compiler until they are a part of string constant. White Space may be used to separate words, but are strictly prohibited while using between characters of keywords or identifiers.

C Character set table

The characters in C are grouped in to 4

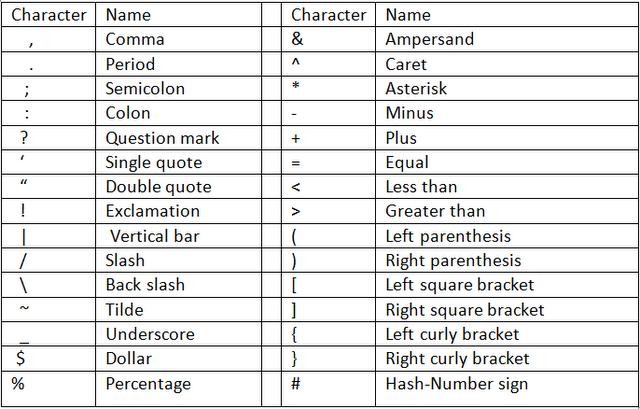
1. Letters(A-Z, a-z)

2. Digits (0-9)

3. Special Characters

4. White spaces(used to separate words)

Special Characters



Escape sequences

1.BlankSpace(\b)

2.HorizontalTab(\t)

3.CarriageReturn(\r)

4.NewLine(\n)

5. Form Feed (\f)

6. Vertical Tab (\v)

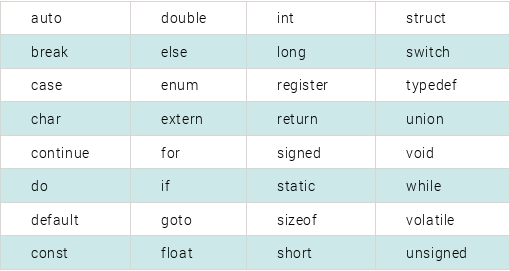
Department of CSE, MEAEC 15

Programming in C Lab Manual **EST 102**

Keywords and Identifiers

Every word in C language is a keyword or an identifier. Keywords are reserved words that have standard, predefined meanings in C. All keywords must be written in lowercase. Keywords in C language cannot be used as a variable name. They are specifically used by the compiler for its own purpose and they serve as building blocks of a c program.

The following are the 32 Keywords of C language.



Identifier refers to the name of user-defined variables, array and functions. A variable should be essentially a sequence of letters and or digits and the variable name should begin with a character.

Both uppercase and lowercase letters are permitted. The underscore character is also permitted in identifiers. The underscore (\_) symbol can be used as an identifier.

Some examples of identifiers are: tax\_rate, \_temp, place etc.

Data Types

Data types are used to store various types of data that is processed by program. The definition of a variable will assign storage for the variable and define the type of data that will be held in the location.

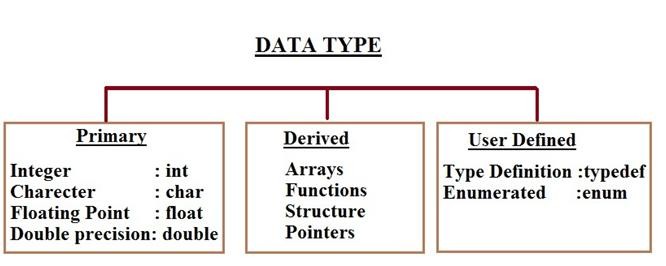
C has different data types for different types of data and can be broadly classified as:

Department of CSE, MEAEC 16

Programming in C Lab Manual **EST 102**

1. Primary data types

2. Secondary data types



Primary data types available in c are

|  |  |  |
| --- | --- | --- |
| Data Type | Description | Memory Requirements |
| int | integer quantity | 2 bytes or 1 word |
| float | floating point number | 1 word(4 bytes) |
| char | single character | 1 byte |
| double | double precision number | 2 words(8 bytes) |

Integer Data Type:-

Integer data types are used to define integer numbers. Integers are whole numbers with a range of values supported by a particular machine. Generally an integer occupies 2 bytes memory space and its value range limited to -32768 to +32767 (that is, -215 to +215-1).

A signed integer use one bit for storing sign and rest 15 bits for number. To control the range of numbers and storage space, C has three classes of integer storage namely short int, int and long int. All three data types have signed and unsigned forms.

A short int requires half the amount of storage than normal integer. Unlike signed integer, unsigned integers are always positive and use all the bits for the magnitude of the number. Therefore the range of an unsigned integer will be from 0 to 65535. The long integers are used to declare a longer range of values and it occupies 4 bytes of storage space.

Syntax: int <variable name>; Examples are:

int num1;

short int num2;

long int num3;

Department of CSE, MEAEC 17

Character Type:

Character type variable can hold a single character. As there are singed and unsigned int (either short or long), in the same way there are signed and unsigned chars; both occupy 1 byte each, but having different ranges. Unsigned characters have values between 0 and

255, signed characters have values from –128 to 127.

Syntax: char <variable name>;

char ch = ‘a’;

Floating Point and Double Types:

The float data type is used to store fractional numbers (real numbers) with 6 digits of precision. Floating point numbers are denoted by the keyword float. When the accuracy of the floating point number is insufficient, we can use the double to define the number. The double is same as float but with longer precision and takes double space (8 bytes) than float. To extend the precision further long double can be used which occupies 10 bytes of memory space.

Syntax: float <variable name>; like float num1;

double num2;

long double num3;

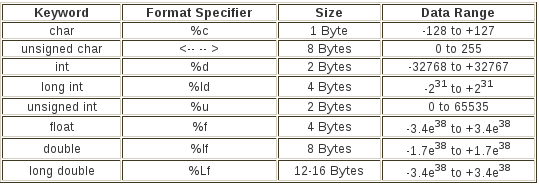
Example: 9.125, 3.1254

Void Type:

The void type has no values therefore it cannot be declared as a variable. The void data type is usually used with function to specify its type. Like in our first C program we declared “main()” as void type because it does not return any value. The

concept of returning values will be discussed in detail in the C function hub.

Data Types in C, Size & Range of Data Types



Constants

A constant is a number, character, or a character string that can be used as a value in a program. Use constants to represent floating-point, integer, enumeration, or character values that cannot be modified. Constants are fixed values that do not change during the execution of a program. C has 4 basic types of constants.

Integer constant

Integer constants are a sequence of digits. It can be written in 3 different number systems

(i) Decimal Integer Constant:

● 0 to 9

● E.g: 49, 58, -62, … (40000 cannot come bcoz it is > 32767)

(ii) Octal Integer Constant:

● 0 to 7

● Add “0” before the value.

● Eg.: 045, 056, 067

(iii) Hexadecimal Integer:

● 0 to 9 and A to F

● Add 0x before the value

● E.g: 0x42, 0x56, 0x67

Programming in C Lab Manual **EST 102**

Floating point constants

Floating point constants are a sequence of digits, followed by a decimal point, followed by a sequence of digits, and optionally followed by an exponent. Ex: +867.9, -26.9876,

654.0 .In exponential form, the real constant is represented as two parts. The part lying before the ‘e’ is the ‘mantissa’, and the one following ‘e’ is the ‘exponent’. For example, the value 237.95 may be written as 23795E2 in exponential notation. E2 means multiply by

102.

Character constants

Character constants are a single character surrounded by single quotes (`''), or a number-

-the ordinal value of the corresponding character (usually its ASCII value). Within quotes, the single character may be represented by a letter or by "escape sequences. Example of character constants are ‘3’,’c’. Since each character constant represents an integer value, it is also possible to perform arithmetic operations on character constants.

String constants

String constants are a sequence of character constants surrounded by double quotes (`"'). The character may be letters, numbers, special characters and blank space. Examples are “hello”,”65”. Each string constant always ends with a special character ‘\0’. The compiler automatically places a null character (\0) at the end of every string constant.

Variables

A variable is a value that can change any time. It is a memory location used to store a data value. A variable name should be carefully chosen by the programmer so that its use is reflected in a useful way in the entire program. Variable names are case sensitive. Example of variable names are,

Sun number Salary Emp\_name average1

Any variable declared in a program should confirm to the following

1. They must always begin with a letter, although some systems permit underscore as

first character.

2. The length of a variable must not be more than 8 characters.

3. White space is not allowed

4. A variable should not be a Keyword

5. It should not contain any special characters.

Variable declaration

The declaration of variables should be done in the declaration part of the program. These variables must be declared before they are used in the program. The declaration provides two things:

1) Compiler obtains the variable name

2) It tells the compiler, the data type of the variable being declared and helps in allocating memory.

3) The syntax of declaring a variable is as follows data\_type variable\_name;

Here data\_type must be a valid data type and variable\_name is an identifier. Some declarations are

int tax;

float tax\_rate,count;

Arrays

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.That means that, for example, we can store 5 values of type int in an array without having to declare 5 different variables, each one with a different identifier. Instead of that, using an array we

Department of CSE, MEAEC 21

can store 5 different values of the same type, int is an example of a unique identifier. For example, an array to contain 5 integer values of type int called temp could be represented like this:

*temp*



Where each blank panel represents an element of the array of type integer values. These elements are numbered from 0 to 4 since in arrays the first index is always 0, independently of its length.

Declaring Arrays

An array is declared by specifying its data type, name and the number of elements the array holds between square brackets immediately following the array name.

Here is the syntax:

*data\_type array\_name[size];*

For example, to declare an integer array which contains 100 elements we can do as follows:

int a[100];

There are some rules on array declaration. The data type can be any valid C data types including structure and union. The array name has to follow the rule of variable and the size of array has to be a positive constant integer.

Array elements can be accessed via indexes array\_name[index]. Indexes of array starts from 0 not 1 so the highest elements of an array is array\_name[size-1].

Initializing Arrays

It is like a variable, an array can be initialized. To initialize an array, provide initializing values which are enclosed within curly braces in the declaration and placed following an equals sign after the array name. Here is an example of initializing an integer array.

int a[3]={1,2,3};

Department of CSE, MEAEC 2

Array and Pointer

Each array element occupies consecutive memory locations and array name is a pointer that points to the first element. Beside accessing array via index we can use pointer to manipulate array. This program helps you visualize the memory address each array elements and how to access array element using pointer.

Multidimensional Arrays

An array with more than one index value is called a multidimensional array. All the array above is called single-dimensional array. To declare a multidimensional array you can do follow syntax

datatype arrayname[size][size][size]

The number of square brackets specifies the dimension of the array. For example to declare two dimensions integer array we can do as follows:

int matrix[3][3];

Statements in C

The statements of a C program control the flow of program execution. In C, as in other programming languages, several kinds of statements are available to perform loops, to select other statements to be executed, and to transfer control. C statements consist of tokens, expressions, and other statements. A statement that forms a component of another statement is called the "body" of the enclosing statement.

Frequently the statement body is a "compound statement." A compound statement consists of other statements that can include keywords. The compound statement is delimited by braces ({ }). All other C statements end with a semicolon (;). The semicolon is a statement terminator.

The expression statement contains a C expression that can contain the arithmetic or logical operators introduced in [Expressions and Assignments](http://msdn.microsoft.com/en-us/library/xw77s4e0.aspx). The null statement is an empty statement. Any C statement can begin with an identifying label consisting of a name and a colon.

C Operator Precedence and Associativity

|  |  |  |
| --- | --- | --- |
| Operator | Description | Associativity |
| () []  .  ->  ++ -- | Parentheses (function call) (see Note 1) Brackets (array subscript)  Member selection via object name  Member selection via pointer  Postfix increment/decrement (see Note 2) | left-to-right |
| ++ --  + -  ! ~ (*type*)  \*  &  sizeof | Prefix increment/decrement  Unary plus/minus  Logical negation/bitwise complement  Cast (change *type*) Dereference Address  Determine size in bytes | right-to-left |
| \* / % | Multiplication/division/modulus | left-to-right |
| + - | Addition/subtraction | left-to-right |
| << >> | Bitwise shift left, Bitwise shift right | left-to-right |
| < <=  > >= | Relational less than/less than or equal to  Relational greater than/greater than or equal to | left-to-right |
| == != | Relational is equal to/is not equal to | left-to-right |
| & | Bitwise AND | left-to-right |
| ^ | Bitwise exclusive OR | left-to-right |
| | | Bitwise inclusive OR | left-to-right |
| && | Logical AND | left-to-right |
| || | Logical OR | left-to-right |
| ?: | Ternary conditional | right-to-left |
| =  += -=  \*= /=  %= &= | Assignment  Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment | right-to-left |