Structured programming

Structured programming

- Although the principles of structured programming have had a profound effect on the programming world, it was not until the 1970s that an actual language was created for teaching structured programming.
- Pascal was developed especially for this purpose. Later generation languages such as C, FORTRAN, and COBOL among others became fully-fledged structured programming languages. The three structures allowed in structured programming are *sequence*, *selection*, and *iteration*.

C program

- C is an structured/procedural systems implementation language.
- C is called a high level, compiler language.
- **C** is a general-purpose computer programming language developed between 1969 and 1973 by Dennis Ritchie at Bell telephone Laboratories.

C Program format

A C program includes the following sections

1. Documentation Section
2. Linker Section
3. Definition Section
4. Global Declaration Section
5. Main() Function Section { Declaration section Executable Section }
6. Sub-program Section

    Function 1

    Function 2

    Function3 etc is User defined functions

Documentation Section

- This section consists of a set of comments lines giving the name of the program, the author and other details which the programmer would like to use later.
- Comments starts with /* and ends with */ and enhances readability and understandability.
- Comment lines are not executable statements ie. executed and are ignored by the compiler.

Link Section

- This section provides instructions to the compiler to link functions from the system library.
- C program have predefined functions stored in the C library. Library functions are grouped category-wise and stored in different file known as header files.
- To be able to access the library files it is necessary to tell the compiler about the files to be accessed.
- Instruction Format: **#include<file_name>** Example #include<stdio.h> a standard I/O header file containing standard input and output functions.

Definition Section

- This section allows the definition of all symbolic constants. Statements begin with # sign and do not end with a ; because the statements are compiler directive statements
- Example #define PRINCIPLE 10000
- Symbolic constants are usually written in upper case to distinguish them from lower case variables. Values defined here remain constant throughout the program.

main() Function Section

- The main() function is a special function used by C system to tell the computer where the program starts. Every program must have exactly **one main function**.
- The instructions contained within this function's definition will always be the first ones to be executed in any C program.

The main function section has the following sections.

- Declaration part: where all variables used in the executable section are declared.
- Executable part: consist of the statements to be executed.

There must be a least one statement in the executable part. The two part must be included between the opening { and the closing }. Program execution begins at the opening { and closes at the closing }

Input radius

Area=3.14*radius*radius

Output area

```
#include <stdio.h>
int main()
{ int rad; float area;
 printf (“Enter radius”);
 scanf (“%d”,&rad);
area= 3.14 * rad * rad;
printf (“Area is %f”, area);
}
```

Main() formats

- All the statements between the { and } forms the function body and are the instructions to perform the given task.
- All statements in the Declaration and Executable parts must end with a semicolon **(;)**

Formats of main()

- main()
- int main()
- void main()

NB:

- C is a case sensitive programming language.
- C has a free-form line structure. End of each C statement must be marked with a semicolon.

C Program Tokens

- In a passage text, individual words and punctuations marks are called tokens.
- Similarly in a C program the smallest individual units are known as C Tokens.
- C program has six types of tokens.

C tokens

- Key words/ Reserved words
- Identifiers
- Constants
- Variables
- Operators
- Special symbols

Reserved Words

- Reserved words (occasionally called keywords) are one type of grammatical construct in programming languages.
- These words have special meaning within the language and are predefined in the language formal specifications.
- Reserved words act as the building blocks for program statements.

C reserved words

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

Identifiers

Identifiers refer to the names of variables, functions and arrays.

Rules for Identifiers

1. First letter must be an alphabet (or underscore).
2. Must consist of only letters, digits or underscore
3. Only first 31 characters are significant
4. Cannot be a Keyword.
5. Must not contain white spaces

Constants

- Constants in C Program refers to fixed values that do not change during the execution of the program. E.g. const pi=3.14;

Variables

- A variable is a named memory location that may take different values at different times during execution.
- A variable name should be chosen in a meaningful way so as to reflect its function.
- Naming of variables should follow the same rules as stated in the identifiers section.

Primary data types

All C compilers support five (5) fundamental data types namely

- Integer (int)
- Character (Char)
- Floating point (float)
- Double-precision floating point (double)
- Void

Integer data types

- Integers are whole numbers with a range of values supported by a particular machine.
- Integer data types are defined in C as **int**.
- C supports three classes of integer storage, **short int, int** and **long int** in both signed and unsigned forms.

Floating point types

- Floating point (or real) numbers are stored in 32 bits (in all 16-bit and 32-bit machines) with 6 digits of precision.
- Floating point data type is defined in C as **float.**
- When the accuracy provided by float is not sufficient, the type double can be used to define the number.

Void types

- Void data type has no values, and usually used to specify the **void type** of function which does not return any value to the calling function example **main(void)**

Character type

- A single character can be defined as character (char) type of data.
- Characters are usually stored in 8-bit (one byte) of internal storage.

Declaring variables

- Variables to be used in a program need to be declared.
- Declaration of variables tells the compiler what the variable name is and specifies what type of data the variable will hold.
- A variable must be declared before it is used in a C program.

Syntax

- Syntax for declaring a variable ***data_type v1, v2, vn;***
- Where **v1, v2** and **vn** are names for variables and the variables are separated with commas.
- A declaration statement must end with a semi-colon.

examples

- int count;
- int count, price; double ratio;

Managing Input, Output and arithmetic Operations

- Reading, processing and writing of data are the three essential functions of a computer program.
- Most programs take data as input and display the processed data, often known as information or output on a suitable medium.

Assignment statement

- The assignment statement implies that the value of the variable on the left of the equal sign is set equal to the value of the quantity on the right.
- Example ***gross_salary = basic_salary + commission;***
- Variables can be assigned an initial value when declaring it in a process called ***initialization***, using the format; ***data type variable_name = constant.***
- E.g. x=60;

Reading data from the keyboard (scanf())

- Another way of assign value to variables is to input data through the keyboard using the ***scanf function***.
- Scanf stands for scan formatted.
- The general format for scanf() is ***scanf("control string", &variable);***
- The control string contains the format of data being received.
- The ampersand **&** before the variable name is an operator that specifies the variable name's address.

control string

%d – integer

%c – character

%f – float and double

%s - string

Example

- Example

scanf(“%d”, &marks);

- When the computer encounters this statement, the programs stops and waits for the value ***marks*** to be keyed in through the keyboard and the <enter key> pressed.
- ***“%d”*** signifies that an integer data type is expected

Arithmetic Expressions

- An arithmetic expression is a combination of variables, constants and operators arranged as per the syntax of the language.
- Expressions are evaluated using an assignment statement of the form **variable = expression;**
- **Example**
- x = a * b – c;
- y = b/c * a;

Operator Precedence

- Operator precedence is used to determine how an expression involving more than one operator is evaluated.
- Most programming language use PEMDAS (Parenthesis, Exponential, Multiplication, Division, Addition and Subtraction) in that order when evaluating an expression.
- To control order of expression evaluation, parenthesis(brackets) are used.

Examples

Evaluate the following:

- Z=x/y*s+x
- Z=x/(y*(s+x))
- where x=18, y=6, s=1,x=2

Displaying results

- The printf function is used to format the C program outputs.
- The printf statement provides certain features that can be effectively exploited to control the alignment and spacing of print-outs on the terminal.
- The printf function format is ***printf(" control string", arg1, arg2,arg3, …..);***
- Example: **Printf ("%d",total);**

Formatted Out put

- The printf function is used to format the C program out puts.
- The printf function format is ***printf(" control string", arg1, arg2,arg3, …..);*** The control string consists of;
- Escape sequence characters such as \n (for new line) and \t (for tab) .
- The argument should match in number, order and type with the format specifications.

Enhancing readability of Output

- Provide enough blank spaces between two numbers.
- Introduce appropriate headings and variable names in the output
- Print special messages whenever a peculiar condition occurs in the output.
- Introduce blank lines between the important sections of the output.

- Printf (“Student name : %s”, stdname);
- Printf (“\n Total marks: %d”, tot);
- Printf(“\n average :%f”, avg);

- Printf (“Student name \t Total \t average);
- Printf (“%s”, stdname, “%d\t”, tot,”%f\t”, avg);