CSE 06131223 ♦ CSE 06131224

Structured Programming

Lecture 6

The Essentials of C Programs (1)

Prepared by

Md. Mijanur Rahman, Prof. Dr.
Dept. of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University, Bangladesh
www.mijanrahman.com

Contents

THE ESSENTIALS OF C PROGRAMS

Basic Structure of C Program

- The structure of a C program means the specific structure to start the programming in the C language.
- Without a proper structure, it becomes difficult to analyze the problem and the solution. It also gives us a reference to write more complex programs.
- A typical C program is a collection of functions, each designed to perform a specific task and return zero or more results to the calling module.
- At a minimum, all C programs must specify one function called ***main***, followed by a left and a right parentheses. A C program may contain one or more sections.

Basic Structure of C Program

Documentation Section

Link Section

Definition Section

Global Declaration Section

Main Function Section

{

Declaration Part

Executable Part

}

Subprogram Section

Function-1

Function-2

….

Function-n

Sample Program:

```
1.  /* This program accepts four integers and calculate and print the average*/
2.  #include <stdio.h>
3.  #include <conio.h>
4.  void main ( )
5.  {
6.  int num1, num2, num3, num4;
7.  float mean;
8.  printf (“Please input 4 integers: “);
9.  scanf (“%d %d %d %d”, &num1, &num2, &num3, &num4 );
10. mean = (num1 + num2 + num3 + num4)/4;
11. printf (“\n 4 numbers are: %d %d %d %d”,num1,num2,num3,num4);
12. printf (“\n The mean of 4 numbers is: %10.2f”, mean);
13. getch();
14. }
```

Programming Style in C

- C is a free-form language. That is, the C compiler does not care, where on the line we begin typing.

Since C is a free-form language, we can group statements together on one line. The statements

```
a = b;
x = y + 1;
z = a + x;
```

can be written on one line as

```
a = b; x = y+1; z = a+x;
```

The program

```
main( )
{
        printf("hello C");
}
```

may be written in one line like

```
main( ) {printf("Hello C")};
```

C Tokens

- In a passage of text, individual words and punctuation marks are called tokens. Similarly, in a C program the smallest individual units are known as C tokens.
- C has six types of tokens. These are:
    1. Keywords
    2. Identifiers
    3. Constants
    4. Strings
    5. Operators
    6. Special Symbols

C Keywords

- Every C word is classified as either a keyword or an identifier.
- The C language reserves certain words that have special meanings to the language. Those reserved words are sometimes called C keywords.
- All keywords have fixed meaning and these meaning cannot be changed.
- All keywords must be written in lowercase.
- You should not use the C keywords as variable, constant, or function names in your program.

C Keywords

- C language has 32 reserved keywords.

Keyword	Description
auto	Storage class specifier
break	Statement
case	Statement
char	Type specifier
const	Storage class modifier
continue	Statement
default	Label
do	Statement
double	Type specifier
else	Statement
enum	Type specifier
extern	Storage class specifier
float	Type specifier
for	Statement
goto	Statement
if	Statement
int	Type specifier
long	Type specifier

Keyword	Description
register	Storage class specifier
return	Statement
short	Type specifier
signed	Type specifier
sizeof	Operator
static	Storage class specifier
struct	Type specifier
switch	Statement
typedef	Statement
union	Type specifier
unsigned	Type specifier
void	Type specifier
volatile	Storage class modifier
while	Statement

C Identifiers

- C words except keyword are identifiers.
- C identifiers represent the name in the C program, for example, variables, functions, arrays, structures, unions, labels, etc.
- An identifier can be composed of letters such as uppercase, lowercase letters, underscore, digits, but the starting letter should be either an alphabet or an underscore.
- If the identifier is not used in the external linkage, then it is called as an internal identifier. If the identifier is used in the external linkage, then it is called as an external identifier.
- **Example of valid identifiers:**

  total, sum, average, _m _, sum_1, etc.

C Identifiers

- **Rules for constructing C identifiers:**
  - The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore.
  - It should not begin with any numerical digit.
  - In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
  - Commas or blank spaces cannot be specified within an identifier.
  - Keywords cannot be represented as an identifier.
  - The length of the identifiers should not be more than 31 characters.
  - Identifiers should be written in such a way that it is meaningful, short, and easy to read.

Keyword vs Identifier

- **Differences between Keyword and Identifier:**

Keyword	Identifier
Keyword is a pre-defined word.	The identifier is a user-defined word
It must be written in a lowercase letter.	It can be written in both lowercase and uppercase letters.
Its meaning is pre-defined in the c compiler.	Its meaning is not defined in the c compiler.
It is a combination of alphabetical characters.	It is a combination of alphanumeric characters.
It does not contain the underscore character.	It can contain the underscore character.

Data Types in C

- **ANSI C supports 4 classes of data types:**
  1. Primary (or fundamental or basic) data types
  2. Derived data types
  3. User-defined data types
  4. Empty (void) data types

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union
Enumeration Data Type	enum
Void Data Type	void

Data Types in C

- **Basic data types:**
- The basic data types are integer-based and floating-point based. C language supports both signed and unsigned literals.
- Four basic data types: char, int, float, double
- The memory size of the basic data types may change according to 32 or 64-bit operating system.

Data Types in C

- **Basic data types:**

Data Types	Memory Size	Range
char	1 byte	−128 to 127
signed char	1 byte	−128 to 127
unsigned char	1 byte	0 to 255
short	2 byte	−32,768 to 32,767
signed short	2 byte	−32,768 to 32,767
unsigned short	2 byte	0 to 65,535
int	2 byte	−32,768 to 32,767
signed int	2 byte	−32,768 to 32,767
unsigned int	2 byte	0 to 65,535
short int	2 byte	−32,768 to 32,767
signed short int	2 byte	−32,768 to 32,767
unsigned short int	2 byte	0 to 65,535
long int	4 byte	-2,147,483,648 to 2,147,483,647
signed long int	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long int	4 byte	0 to 4,294,967,295
float	4 byte	
double	8 byte	
long double	10 byte	

Variables in C

- A *variable* is a data name that may be used to store a data value. A variable may take different values at different times during execution.
- Thus, a variable is nothing but a name given to a storage area that our programs can manipulate.
- Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.
- The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C is case-sensitive.

Variables in C

- Based on the basic types, there will be the following basic variable types –

Sr.No.	Type & Description
1	**char** Typically a single octet(one byte). It is an integer type.
2	**int** The most natural size of integer for the machine.
3	**float** A single-precision floating point value.
4	**double** A double-precision floating point value.
5	**void** Represents the absence of type.

Variables in C

Variable Declarations:

- Before you can use a variable in a C program, it must be declared. A variable declaration tells the compiler the name and type of a variable and optionally initializes the variable to a specific value.
- A variable declaration has the following form:

  typename varname;

- *typename* specifies the variable type and must be one of the keywords listed in Table 3.2. *varname* is the variable name, which must follow the rules mentioned earlier. You can declare multiple variables of the same type on one line by separating the variable names with commas:

```
int count, number, start;    /* three integer variables */
float percent, total;        /* two float variables */
```

Variables in C

- **Assigning Values to Variables:**
- Values can be assign to variables using the assignment operator = as follows:

  Variable_name = constant_value;
- For example:

  N = 100;

  X = 75.50;

  Yes = 1;

Variables in C

- **Initialization:**
- The process of giving initial values to variables is called initialization.

  Data-type variable_name = constant;
- For example:

  int n = 10;

Also,

x = y = z = 0;

is valid.

THE END