Paper 102:Programming & Problem solving through C

Lecture-01:Unit-I

C Fundamentals

A.V Hujon, Dept. of Computer Science, SAC

Outline of the course

- Theory-100 marks (EA-75,IA-25)
- Practical-100 marks (EA-75,IA-25)

Credit values

- Theory - 4
- Practical - 2

Internal Assessment

- Three Sessional Exams
  - out of 25 marks each.

A brief history of C

- UNIX was developed at around 1969. Its first version was programmed in assembly Language and run on a DEC PDP-7.
- The second version was ported to a PDP-11 in 1971 and it was a great success:
  - 16 KB for the system
  - 8 KB for user programs
  - disk of 521 KB
  - limit of 64 KB per file
- While writing a FORTRAN compiler for UNIX, a new programming language was developed: B
- B was interpreted and, therefore, slow. To solve the performance problems of B, a new language was created:
  - C in 1972 by Dennis Ritchie
- In 1973 UNIX was rewritten in C
- Standardized in 1989 by ANSI (American National Standards Institute) known as ANSI C

C as a programming language

The basic characteristics of C are:

1. Small in size
2. Loose typing (lots of freedom, error prone)
3. Structured (extensive use of functions)
4. Designed for systems programming (i.e., Low level programming of the type needed to implement an operating system)
5. C is higher level than assembler but still close to the hardware and allows direct manipulation of many system aspects: pointers, memory allocation, Bitwise manipulation ...

The C character Set

- Alphabets: A-Z, a-z
- Digits: 0-9
- Special Characters: (all the symbols on the keyboard) ~, `, !, @, #, $, %, ^, &, *, (,), - , _, =, +, [, {,], }, \, |, ;, :, ', ", ,, . , <, ., >, /, ?
- White space characters:
  - **Nonprintable characters like blank space, tab space, newline, carriage return**

C Basic Data Types

Data Type	Size in Bytes, possible range of values
char	1 ,-128 to 127
int	2 , -32768 to 32767
float	4 , $3.4 * 10^{-38}$ to $1.7 * 10^{38}$
double	8 , $1.7 * 10^{-38}$ to $3.4 * 10^{38}$

Identifiers and keywords

- Every word in C is either classified as an **Identifier** or a **keyword**.
- **Identifiers**: Use to identify names of variables, symbolic constants, function names, etc.
- **Keywords**: Predefined term having a predefined meaning, which cannot be changed by the user.

Rules for an identifier

- It must be a sequence of letters, digits, and must begin with a letter
- The underscore character (_) is considered as a letter.
- Names should not be the names of keywords.
- C is case-sensitive
- The first 31 characters of an identifier's name is significant in ANSI C compiler.

Variables

- A variable name is an entity that has a value stored in memory, and the location is known to the program by a name.
- Rules that apply to identifier applies to variables also.
- E.g., emp_number, marks, MAX, _fact, number1
- Every variable must be created to store a specific type of data (whole numbers, numbers with a fraction, strings etc.) or data types.

Arithmetic Operators

There are five arithmetic operators in C

Operator	Description	Example
+	add	2 + 3
-	subtract	Number – 1
*	multiply	density1*2.5
/	division	density1/density2
%	remainder after integer division (modulus)	Number % 2

Assignment Operators

Operator	Description	Example
=	Assignment	Number = 6;
+=	Sum and assign	Number += 10; adds 10 to Number
-=	Subtract and assign	Number -= 5; subtract 5 from Number
*=	Multiply and assign	density1 *=Number; same as density1=density1*Number;
/=	Divide and assign	density1/=4; divides density1 by 4
&=	bitwise AND and assign	var1&=var2 ands the bits of var1 with var2
|=	bitwise OR and assign	var1|=var2 ors the bits of var1 with var2

Logical and Equality Operators

There are two logical operators in C

&&	AND
||	OR

There are six equality operators

C Operator	Description
==	Equal to =
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Unary Operators

A unary operator acts upon a single operand to produce a new value.

1. Unary Minus

   -3

2. The logical not operator is also frequently used.

   !

   C also offers a bitwise not operator, exchanging 0s and 1s in the binary representation.
   This operator is represented by the tilde symbol ~.

3. Unary Pointer Operators

C Operator	Description
&	Address of
*	Indirection

4. Increment and Decrement

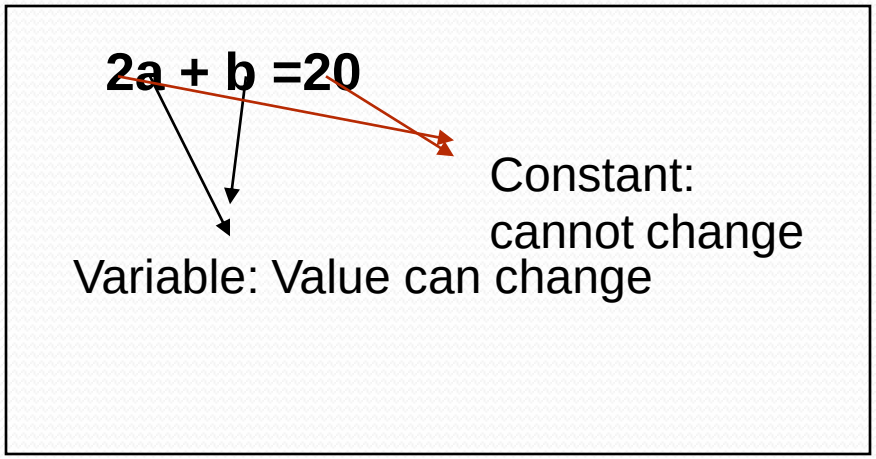
C has unary operators for both increment and decrement.

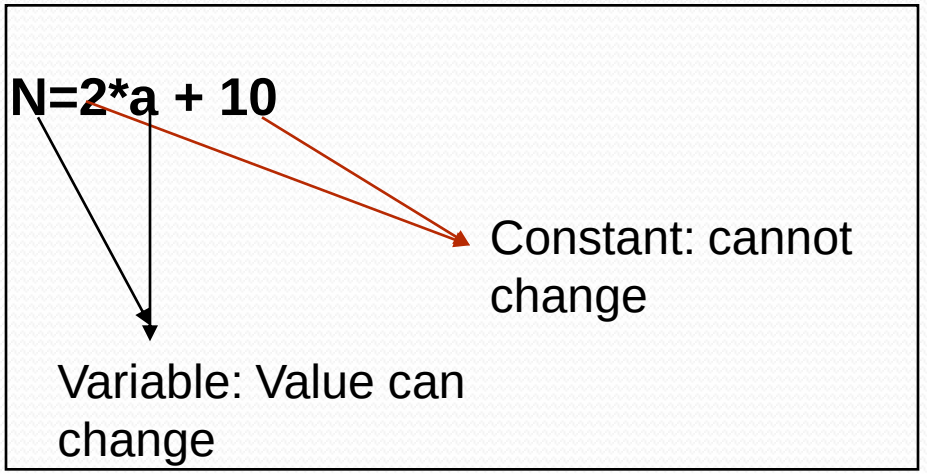
Operator	Description	Example
++	increment	x++; ++y;
--	decrement	x--;--y;

Reserved Words in C

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

Constants





C constants can be divided into two major categories:

- Integer: 426,+777,-8976
- Real: Two form-Fractional form. 426.06, 10.5. 3.0
  Exponential form. +3.2e+3, 1.5e8
- Character: 'A', 'I', '+', '='

Rules for constructing integer constants

1. It must have at least one digit
2. It must not have a decimal point
3. It can be either positive or negative
4. The default sign is positive
5. No commas, or blanks are allowed within an integer constant
6. The allowable range is -32767 to +32767 (16 bit machine)

Rules for constructing Real constants: Fractional form

1. It must have at least one digit
2. It must have a decimal point
3. It can either be positive or negative
4. Default sign is positive
5. No commas, or blanks are allowed within a real constant

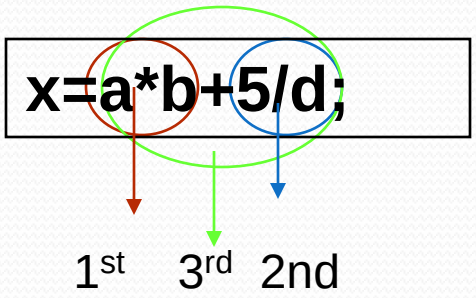
Rules for constructing Real constants: Exponential form

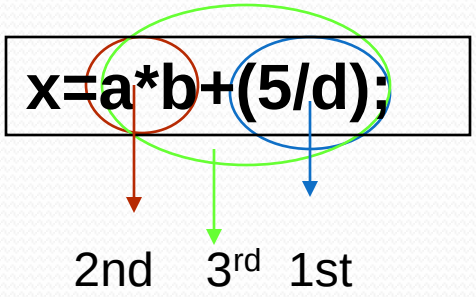
+2.5e5 **+2.5 is called Mantissa, and 5 is called the exponent**

1. The mantissa and the exponent should be separated by the letter e
2. Both can either be positive or negative
3. Both must have at least one digit
4. Default sign is positive
5. Range of real constants is -3.4e38 to 3.4e38

Hierarchy of operations

- 1st priority - * / %
- 2nd priority- + -
- 3rd priority - =





Algebraic expression	C expression
a x b – a x c	a*b-a*b
(a +b) (a + b)	(a+b) * (a+b)
$2x^2 + 2x + 5$	2*x*x + 2*x + 5

C Instructions: rules applicable to all C statements

- Blank spaces may be inserted between two words to improve readability, however no blank spaces are allowed within a variable, constant, or keyword
- Usually all statements are entered in small case letter (all keywords are in small letters)
- C statement always ends with a ; (semicolon)
- Comments are added using /* comments */ for multiple lines and // comment for single lines
- A block of statement are enclosed within { block of codes }

C Instructions

- Type declaration - **to declare the type of variables used in a C program**
- Input/output - **to perform the function of supplying input data to a program and obtaining the output results from it.**
- Arithmetic -**to perform arithmetic operations between constants and variables.**
- Control - **to control the sequence of execution of various statements in a C program**

Type Declaration

```
int  num;
float price;
char ans;

int  num=0;
float price=1.0;
char ans='y';

int  num1,num2;
float price, average;
char ans, response;
```

Anatomy of C Program

- A collection of functions
  - Receive a set of parameters
  - Declare local variables
  - Carry out processing
  - Return a value
- `main()` function
  - Called to start the program

C libraries

- Most programs are not built from scratch
- Rely on pre-existing collections of functions
  - e.g. the Standard C library
- Header (.h) files describe functions in these collections
  - e.g. accessed through **#include** statements

A C function definition

```
type function(argument_list)
  {
  variable_declarations;

  statements;
  }
```

- Each function has a type
- Each function argument has a type
- Each local variable has a type

A simple C program

```
/* C code is stored in .c or .cpp files */

#include <stdio.h>

int main()
  {
  printf("Hello, I am a program\n");

  return 0;
  }
```

Output:

Hello, I am a program

Another C program

```
#include <stdio.h>          /*using one of C libraries*/
void main()
{
    int a,b=5;              /* declaring variables a and b */
    a=b * 10;               /* arithmetic statement*/
    printf("The value of a is %d",a); /* i/o statement */
}
```

Output:

The value of a is 50

Class assignment

- List out 10 valid identifiers and 10 invalid identifiers
- A shopkeeper needs to store information of his grocery shop. List out the appropriate variables and their type required to perform the task. Show the declaration of these variables.