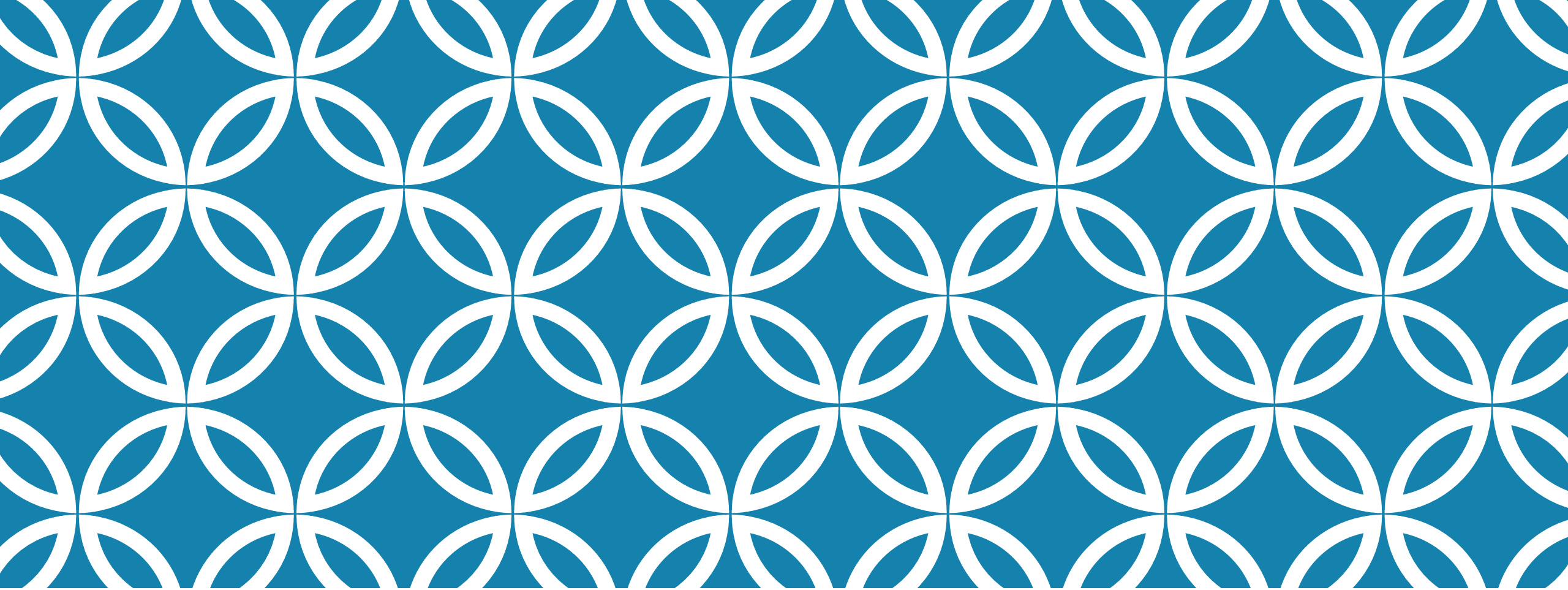


PAPER 102:
PROGRAMMING & PROBLEM SOLVING THROUGH C

Bidur Bhushan Handique
Dept. of Computer Science
St. Anthony's College

BOOKS TO REFER:

1. Let Us C
   - Yashwant Kanetkar
2. Programming in ANSI C
   - E Balaguruswamy

CONTENTS

- Brief History
- Characteristics of C
- Steps of learning C language
- C Character set
- C Data Types
- Constants, Variable and Keywords

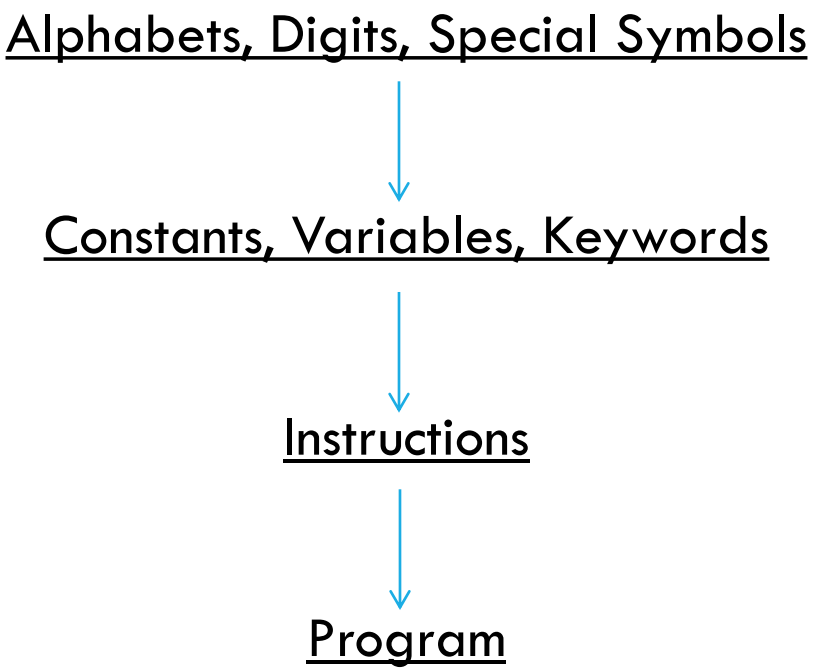
BRIEF HISTORY

- UNIX was developed at around 1969. Its first version was programmed in assembly language and run on a DEC PDP-7.
- While developing a FORTRAN compiler for UNIX, a language known as B was developed.
- But B was slow. And so, C was developed at AT & T's Bell Laboratories of USA in 1972.
- It was designed and developed by Dennis Ritchie.
- UNIX was rewritten in C in 1973.
- Standardized in 1989 by ANSI (American National Standards Institute) known as ANSI C

CHARACTERISTICS OF C

1. It is reliable, simple and easy to use.
2. Structured (extensive use of functions).
3. Portable because software installation is platform independent.
4. Extendable, existing system can be extended i.e. new feature can be added.
5. Designed for systems programming.
6. C is higher level than assembler, but programs written in C run at speed matching to that of the same programs written in assembly language.
7. C allows direct manipulation of many system aspects like pointers, memory allocation and many …

STEPS OF LEARNING C LANGUAGE



C CHARACTER SET

Denotes alphabets, numbers and special symbols.

- Alphabets :
  - A, B, ..., Y, Z (Capital)
  - a, b, …, y, z (Small)
- Numbers :
  - 0, 1, …, 8, 9
- Special Symbols:
  - ~ ! @ # $ % ^ & * ( ) _ = + - * / { } [ ] ; : < > ? . , |' “

C DATA TYPES

Why we need 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 $3.4 * 10^{38}$

CONSTANTS, VARIABLES AND KEYWORDS

CONSTANTS:

- Also known as *Literals*.
- Constants are values that remains unaltered during the execution of a program.
- Constants can be of any data types.
- A constant is a value assigned to a variable defined with any data type.
  - Eg:
    - int x=5, y; y=8;
    - float b=4.5;
- Constant types:
  - Primary Constants:
    - Integer constant, Real Constant, Character Constant
  - Secondary Constants:
    - Array, Pointer, Structure, Union

CONTINUED

Rules for constructing integer constants:

- It must have at least one digit.
- It must not have a decimal point.
- It can be either positive or negative.
- The default sign is positive.
- No commas, or blanks are allowed within an integer constant.
- The allowable range is -32767 to +32767 (16 bit machine).
  - Eg:
    - 1
    - -23

CONTINUED

Rules for constructing real constants:

- It must have at least one digit.
- It must have a decimal point.
- It can either be positive or negative.
- Default sign is positive.
- No commas, or blanks are allowed within a real constant.
  - Eg:
    - .1
    - -45.6

CONTINUED

Rules for constructing real constants (Exponential Form):

+2.5e5, here +2.5 is called the Mantissa and 5 is called the Exponent

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

CONTINUED

VARIABLES:

- An entity that may vary during program execution.
- Variables are names assigned to memory locations.
- A variable can store a value based on the data type used to define the variable.
- General syntax of defining a variable:
  - Data type <Variable_list>;
  - Eg:
    - int a;
    - float a, b, c;
- Defining a variable means to tell a compiler where and how much space is needed to be created in the memory.

CONTINUED

Rules for constructing Variable names:

- Variable names are combination of alphabets, numbers and underscore '_' .
- A variable name should be at the most 31 characters long, but some compilers allow 247 characters long.
- First character of any variable name should be a character or underscore.
- No commas or spaces are allowed within a variable name.
- No special symbols other than underscore are allowed.
  - Eg:
    - A
    - A_B
    - _C

IDENTIFIERS AND KEYWORDS

- Every word in C is either an IDENTIFIER or a KEYWORD.
- Identifier:
  - User defined.
  - Use to identify names of variables, symbolic constants, function names, etc.
    - Eg:
      - int sum; //here sum is an identifier
- Keyword
  - Also known as 'Reserved Words'.
  - Pre defined term.
  - Having a predefined meaning in C compiler, which cannot be changed by the user.
  - And so we cannot use a keyword as variable names.
  - The list of 32 keywords are as follows:
    - 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

OPERATORS

- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.
- Following are the types of operators C provides:
  - Arithmetic Operators
  - Assignment Operators
  - Logical and Equality Operators
  - Unary Operators

CONTINUED

ARITHMETIC OPERATORS:

OPERATOR	DESCRIPTION	EXAMPLE
+	Addition	2+3
-	Subtraction	3-2
*	Multiplication	3*2
/	Division (Computes quotient)	3/2
%	Modulus (Computes remainder)	3%2

CONTINUED

ASSIGNMENT OPERATORS:

OPERATOR	DESCRIPTION	EXAMPLE
=	Assignment	Number = 10;
+=	Sum and Assign	Number += 10;//adds 10 to number
-=	Subtract and Assign	Number -= 10;//subtract 10 from number
*=	Multiply and Assign	Number *= 10;//Number = Number * 10;
/=	Divide and Assign	Number /= 10;//Number = Number / 10;
&=	Bitwise AND and Assign	Letter1&=Letter2;//ands the bits of Letter1 with Letter2
|=	Bitwise OR 2 and Assign	Letter1|=Letter2;// ors the bits of Letter1 with Letter2

CONTINUED

LOGICAL OPERATORS

OPERATOR	DESCRIPTION
&&	AND
||	OR

EQUALITY OPERATORS

OPERATOR	DESCRIPTION
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than equal to
>=	Greater than equal to

CONTINUED

UNARY OPERATORS

These operators acts upon a single operand to produce a new value.

1. Unary Minus

```
-3
```

2. The logical not operator is also frequently used.

```
!(x>y)
```

3. Pointer Operators

C Operator	Description
&	Address of
*	Indirection

CONTINUED

4. Increment and Decrement Operators

OPERATOR	DESCRIPTION	EXAMPLE
++	Increment	i++; ++i;
--	Decrement	i--; --i;

5. Sizeof: The sizeof operator returns the size of its operand in bytes.

```
int I; printf("Size of integer : %d",sizeof(i));
```

6. Bitwise NOT Operator

Is represented by tilde ~

Eg:

```
unsigned short y = 0xAAAA;
y = ~y;
```

Here, the new value assigned to y is the one's complement of the unsigned value 0xAAAA

HIERARCHY OF OPERATIONS

PRIORITY	OPERATORS	DESCRIPTION
1st	*, /, %	Multiplication, Division, Modular Division
2nd	+, -	Addition, Subtraction
3rd	=	Assignment

Note:

An operation within parenthesis will have the highest priority.

Eg:

x = a * b + b / d;

i) Multiplication
ii) Division
iii) Addition

x = a * b + (a / b);

i) Division
ii) Multiplication
iii) Addition

WAY TO CONVERT GENERAL ALGEBRAIC EXPRESSION TO A 'C' EXPRESSION

ALGEBRAIC EXPRESSION	'C' EXPRESSION
$a \times b - c \times d$	a * b – c * d
(m + n) (a + b)	(m + n) * (a + b)

RESERVED WORDS

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

C INSTRUCTIONS

- 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
  - // for single line
  - /* comment */ for multiple lines
- A block of statement are enclosed within { }
- C is case sensitive.
  - Eg.:
    - int a, A; // are different

C INSTRUCTIONS

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

ANATOMY/STRUCTURE OF C PROGRAM

A collection of function

- Receive a set of parameters
- Declare local variables
- Carry out processing
- Return a value

main() function

- Called to start the program

C LIBRARIES

- Basically C programs are not built from scratch.
- Rely on pre-existing collections of functions
  - Eg:
    - The Standard C Library
- The C Standard Library is a set of C built-in functions, constants and header files like <stdio.h>, <conio.h>, <math.h>, etc.
- printf, scanf are some functions…

A SIMPLE C PROGRAM

```
// WAP in C to print “Hello World”.
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf(“\n Hello World \n”);
getch();
}
```

TRY THESE PROGRAM IN LAB

Firstly write an algorithm and then write the programs of the following:

- Hello World
- Print Integer
- Addition
- Odd or even

Note: Take care of indentation when you program.

COMPILE AND EXECUTE

Follow the steps to compile and execute:

- Start the compiler at **C>** prompt. The compiler (TC.EXE is usually present in **C:\TC\BIN** directory).
- Select **New** from the **File** menu.
- Type the program.
- Save the program using **F2** under a proper name (say Program1.c).
- Use **Ctrl + F9** to compile and execute the program.
- Use **Alt + F5** to view the output.