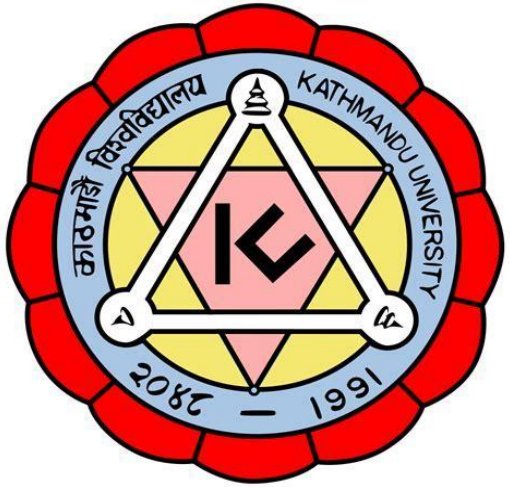
KATHMANDU UNIVERSITY SCHOOL OF MANAGEMENT

BBIS

COM 102 : 3 Credit Hours



1. Introduction to C

25/12/2021

Outlines

1.1 History of C

1.2 Introduction to C

1.3 Types of Programming Languages,

1.4 Basic Structure of C program

1.5 Types of errors

History of Programming

- The root of all modern languages is ALGOL (introduce in 1960s).
- ALGOL uses a structure programming.
- ALGOL is popular in Europe.
- In 1967, Martin Richards developed a language called BCPL (Basic Combined Programming Language).
- Primarily, BCPL is developed for system software.
- In 1970, Ken Thompson created a new language called B.
- B is created for UNIX OS at Bell Laboratories.
- Both BCPL and B were "typeless" languages.

1.1 History of C

- C programming language was developed in 1972 by Dennis Ritchie at Bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.
- C was evolved from ALGOL, BCPL and B.
- Added new features and concepts like "data types".
- It was developed along and strongly integrated with the UNIX operating system.
- In 1983, American National Standards Institute (ANSI) appointed a technical committee to define a standard for C.
- The committee approved a version of C in December 1989 which is now known as ANSI C.
- In 1990 International Standards Organization (ISO) has approved C and this version of C is referred to as C89.

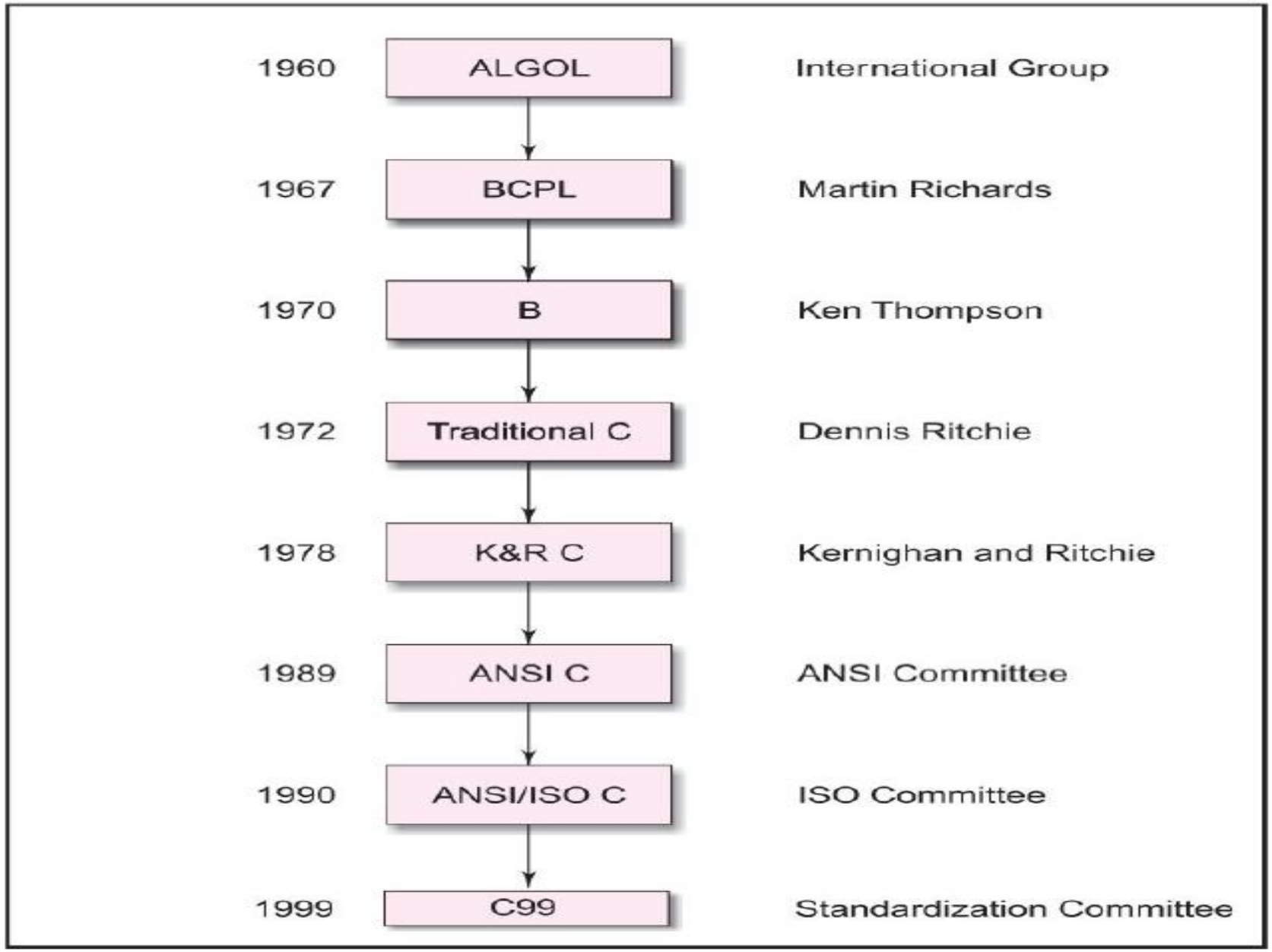


Fig. 1.1 *History of ANSI C*

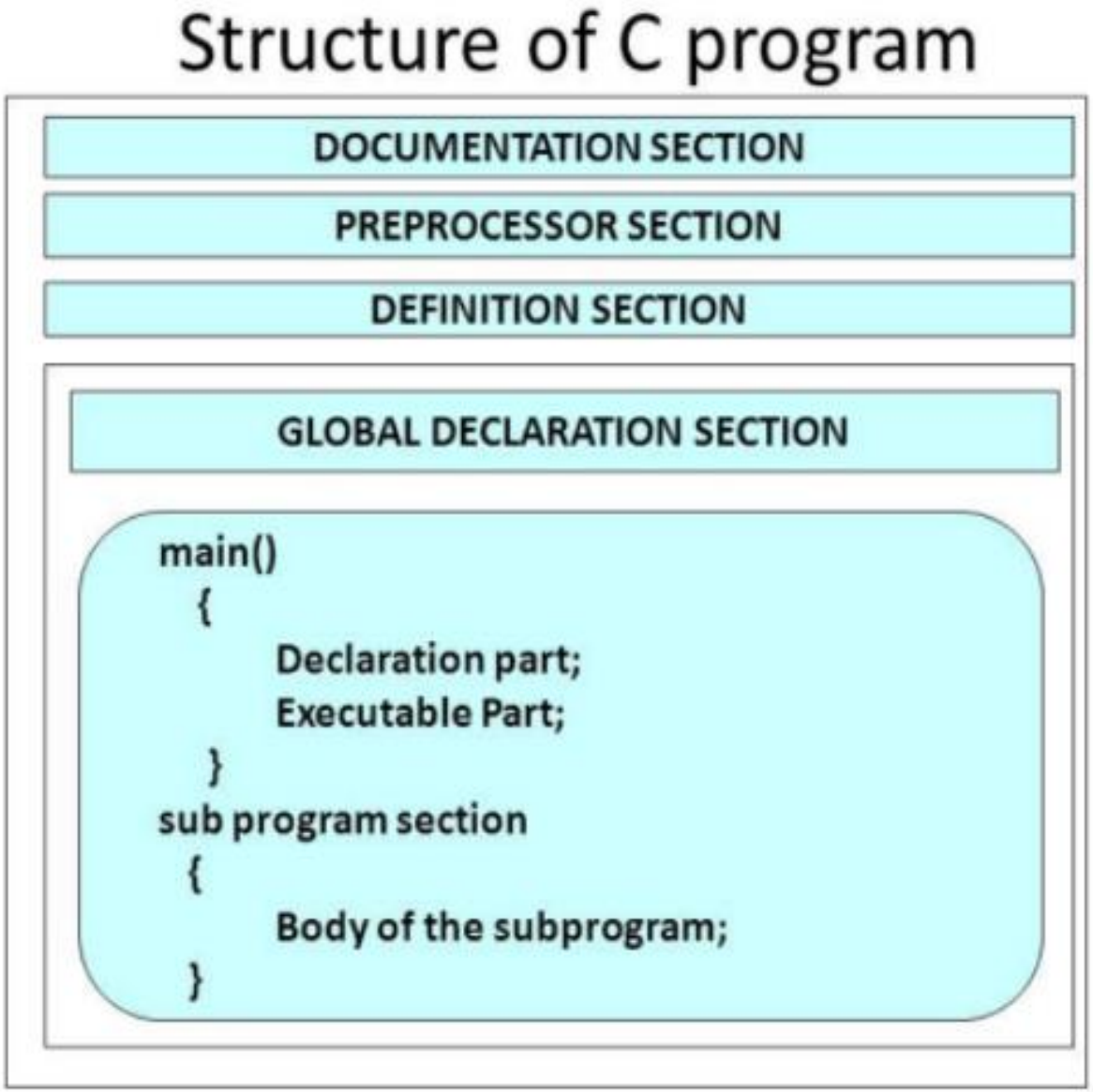
Key advantages of learning C Programming

- C is the most widely used computer programming language.

Some of the key advantages of learning C Programming:

- Easy to learn
- Reliable
- Simple
- Structured language
- It produces efficient programs
- It can handle low-level activities
- It can be compiled on a variety of computer platforms

1.4 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

Basic C Program Structure …

Documentation Section: The line /*...*/ or // will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.

Preprocessor/Link Section: line of the program #include <stdio.h> is a preprocessor

command, which tells a C compiler to include stdio.h file before going to actual compilation.

Definition Section: The definition section defines all symbolic constants. A symbolic constant is a constant value given to a name which can't be changed in program. Example: #define PI 3.14

> **IMP:** Do NOT put a semicolon character at the end of #define statements. This is a
>
> common mistake.

Global Declaration: In global declaration section user defined functions are declared.

Example: int add(int a, int b);

int x;

Basic C Program Structure …

main() Function Section: The main () function section is the most important section of any C program. The compiler start executing C program from main() function. The main() function is mandatory in C programming. It has two parts:

Declaration Part - All the variables that are later used in the executable part are declared in this part.

Executable Part - This part contains the statements that are to be executed by the compiler.

Subprogram Section: The subprogram section contains all the user defined functions. A complete C program can be written without use of user-defined functions but the code maybe be redundant and inefficient if user-defined functions are not used for complex programs.

```c
/* Program:  Area Of Circle
   Author: Alien  */
#include<stdio.h>
#include<conio.h>

#define PI  3.14

void area(int);

main()
{
   int radius;
   printf("Enter Radius Of Circle ");
   scanf("%d",&radius);
   area(radius);
}

void area(int r)
{
   float result;
   result = PI*r*r;
   printf("Area Of Circle is %f", result);
}
```

Documentation Section

Link Section

Definition Section

Global Declaration Section

Main() Function Section

Subprogram Section

1.5 Types of errors

- Basically there are three types of errors in c programming:
  1. Runtime Errors
  2. Compile Errors
  3. Logical Errors

1. Runtime Errors

Runtime errors are those errors that occur during the execution of a program and generally occur due to some illegal operation performed in the program.

Examples of some illegal operations that may produce runtime errors are:

- Dividing a number by zero
- Trying to open a file which is not created
- Lack of free memory space

A program should be written such that it is able to handle such unexpected errors and rather than terminating unexpectedly, it should be able to continue operating.

- This ability of the program is known as **robustness**

2. Compile Errors

- Compile errors are those errors that occur at the time of compilation of the program.
- C compile errors may be further classified as:
  - Syntax Errors
    - When the rules of the c programming language are not followed, the compiler will show syntax errors.
    - Eg: int a,b:
  - Semantic Errors
    - Semantic errors are reported by the compiler when the statements written in the c program are not meaningful to the compiler.
    - b+c=a; Incorrect
    - a=b+c; Correct

3. Logical Errors

- Logical errors are the errors in the output of the program. The presence of logical errors leads to undesired or incorrect output and are caused due to error in the logic applied in the program to produce the desired output.

- Also, logical errors could not be detected by the compiler, and thus, programmers has to check the entire coding of a c program line by line.

References

- https://clanguagebasics.com/types-of-errors-in-c/
- https://www.tutorialspoint.com/errors-in-c-cplusplus