Chapter: 3
Basic Concepts of C

Advanced College of Engineering and Management

Topics to be covered

- Introduction
- Basic Structure of C Program
- Compilation process
- Character Sets
- Keywords
- Identifier
- Data Types
- Variables
- Constants
- Preprocessor Directives
- Symbolic Constants
- C Tokens

Introduction

- C was Introduced by Dennis Ritchie.
- At Bell Lab in 1972.
- Follows top down approach.
- Case sensitive – use lower case letter

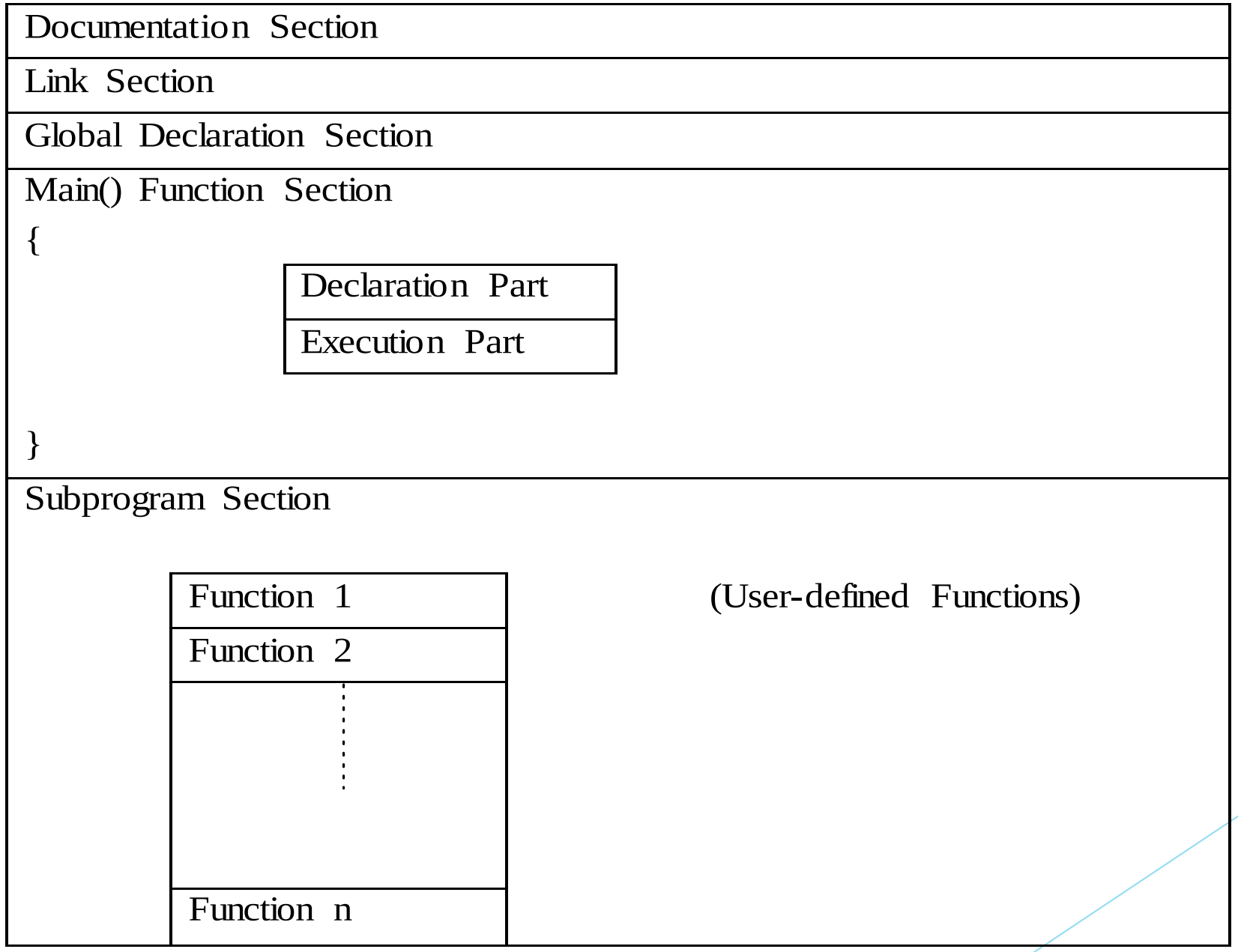
History of C

Language	Year	Developed By
Algol [Algorithmic Programming]	1960	International Group
BCPL[Basic Combined Programming Language]	1967	Martin Richard
B	1970	Ken Thompson
Traditional C	1972	Dennis Ritchie
ANSI C [American National Standard Institute]	1989	ANSI Committee
ANSI/ISO C [International Organization for Standardization]	1990	ISO Committee
C99	1999	Standardization Committee

Features of C

- It is a robust language with rich **set of built-in functions** and **operators** that can be used to write any complex program.
- The C compiler combines the **capabilities of an assembly language** with **features of a high-level language.**
- Programs Written in C are **efficient and fast**.
- C is **highly portable** this means that programs **once written can be run on another machines** with little or no modification.
- Another important feature of C program, is its ability to **extend** itself.
- A C program is basically a collection of functions that are supported by C library. We can also create our **own function** and add it to C library.
- C language is the most widely used language in operating systems and embedded system development today.

Structure of C Program



Example

```
// Program to add two numbers
#include<stdio.h>
#include<conio.h>
int sum(int,int);
int s;
void main()
{
      int a,b;
      a=2;
      b=3;
      s=sum(a,b);
      printf("Sum is %d",s);
      getch();
}
int sum(int x, int y)
{
      return(x+y);
}
```

Document Section

Link Section

Global Declaration Section

Declaration Part

Execution Part

Main Function Section

User Defined Function

Character Sets

- **Alphabets**

  -- Uppercase letters A - Z

  -- Lowercase letters a – z

- **Digits**

  -- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- **Special Characters**

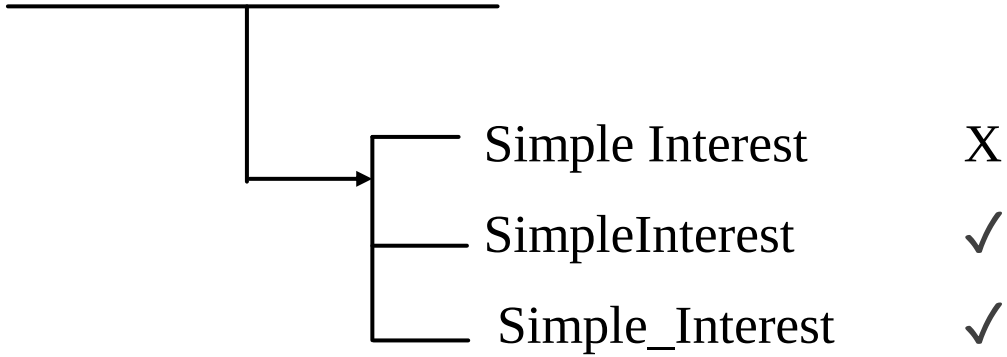
  ~ → tilde
  | → vertical bar
  + → plus sign
  _ → underscore
  > → greater than
  % → percent sign
  @ → at symbol
  < → less than
  - → minus sign
  ^ → caret etc…..

- **White Space**

  \b → blank space
  \v → vertical tab
  \f → form feed
  \t → horizontal tab
  \r → carriage return
  \n → new line etc…

Identifiers

- Identifiers are names that are given to various program elements, such as variables, functions and arrays.
- Rules for identifiers
  - First character must be an alphabet (or Underscore).
  - Must consist of only letters, digits or underscore.
  - Only first 31 characters are significant.
  - Cannot use a keyword.
  - Must not contain white space.



Keywords

- There are certain **reserved** words, called keywords that have standard, predefined meanings in C.
- They cannot be used as programmer-defined identifiers.
- There are 32 keywords in c programming they are:

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			

Data types

- The data type in **C** defines the amount of storage allocated to variables, the values that they can accept, and the operation that can be performed on those variables
- There are following type of data types supported by c programming

1. Primary Data Type
2. Derived Data Type
3. User Defined Data Type

Primary(Fundamental) Datatypes

- Character type
- Integers type
- Floating type
- Void type

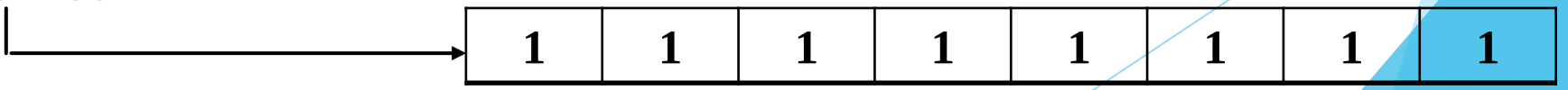
Character Type

- A single character can be defined as a character type data.
- Characters are stored in 8 bits (1 Byte).
- It is represented by a keyword “char”.
- Its conversion character is %c.

Type	Memory Required	Conversion Character	Value Range
Char	1 byte	%c	-128 to 127 i..e -2^7 to 2^7-1
Unsigned char	1 byte	%c	0 to 255 i..e 0 to 2^8-1
Signed char	1 byte	%c	-128 to 127 i..e -2^7 to 2^7-1

- Note 1 Byte= 8 bit
- Largest possible value that can be represented in 8 bit is (2^8-1) i..e 255

Binary of 255



Integer Type

- ✓ Integers are the whole numbers, i.e. non-fractional numbers.
- ✓ Integers are stored in 16 bits (2 Byte).
- ✓ Generally it is represented by a keyword “int”.
- ✓ Integer are of following type.

Type	Represented by	Conversion Character	Memory occupied	Value Range
Signed Integer	signed int or int	%d	2 byte	-2^{15} to $2^{15}-1$
Unsigned Integer	unsigned int	%u	2 byte	0 to $2^{16}-1$
Signed Short Integer	signed short	%d or %i	2 byte	-2^{15} to $2^{15}-1$
Unsigned Short Integer	unsigned short	%u	2 byte	0 to $2^{16}-1$
Signed Long Integer	signed long int or long int	%ld	4 byte	-2^{31} to $2^{31}-1$
Unsigned Long Integer	unsigned long int	%lu	4 byte	0 to $2^{32}-1$

Floating Point Types

- ✓Floating types are fractional numbers (i.e. real numbers).
- ✓They are defined in c by keyword float.
- ✓Generally floating numbers reserve 32 bits of storage.

Type	Represented by	Conversion Character	Memory Occupied	Value Range
Floating Point	Float	%f	4 byte	-2^{31} to $2^{31}-1$
Double	double	%lf	8 byte	-2^{63} to $2^{63}-1$
Long Double	long double	%Lf	10 byte	-2^{79} to $2^{79}-1$

Void Type

- ✓The void type has no value.
- ✓It is usually used to specify type of function when it does not return any value to calling function

User Defined Data types

- The user defined data types enable a program to invent his own data types and define what values it can take on.
- C supports 2 types of user defined data types.
  - typedef (type definition)
  - enum (enumerated data type)

Example of typedef

Program without using typedef	Program using typedef

#include<stdio.h>
void main()
{
int a;
a=3;
printf("Value of a=%d",a);
}
 	
	#include<stdio.h>
	void main()
	{
	typedef int num;
	num a;
	a=3;
	printf("Value of a=%d",a);
	}

Enumeration

- An **enumeration** is used in any programming language to define a constant set of values. For **example**, the days of the week can be defined as an **enumeration** and used anywhere in the program
- Example:

```
#include <stdio.h>
enum week {Sunday=1, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};
int main()
{
   // creating today variable of enum week type
   enum week today;
   today = Sunday;
   printf("Day %d",today);
   return 0;
}
```

Derived Datatype

- ✓ Data types that are derived from the built-in data types are known as derived data types.
- ✓ The various derived data types provided by C are *arrays*, *pointers and structures*.

Example of Structure

```
struct student
{
    char Name[20];
    int Roll;
    float Marks;
}s;
```

Variables

- variable is an identifier that is used to represent a single data item, i.e., a numerical quantity or a character constant

  Example: *int num*; Here num is an integer type variable.

  float num; Here num is an floating type variable.

The declaration of variables mainly has two significances:

- It gives name of variables and its type.
- It allocates appropriate memory space according to its data types.

Rules for naming variables:

- First character must be an alphabet (or Underscore).
- Must consist of only letters, digits or underscore.
- Only first 31 characters are significant.
- Cannot use a keyword.
- Must not contain white space.

Constants\Literals

- constant is a value or an identifier whose value cannot be altered in a program.

  Example: 1, 2.5, "C programming is easy", etc.

Constants can be

- Integer constant — **const** int a= 5;
- Floating point constant — **const** float b= 3.14;
- Character constant — **const** char a='A';
- String Constant — **const** str[15]="My name is C Programming";

Integer Constant

- Integer constant is a numeric constant (associated with number) without any fractional or exponential part

  Example: Decimal constant :- 2, 0, 9, -20 etc..

  Octal Constant:- 021, 077, 033 etc..

  Hexadecimal constant: - 0x7F, 0x2A etc

Floating-point constants

- A floating point constant is a numeric constant that has either a fractional form or an exponent form.

  Example:- 2.0, 0.0000234, -0.22E-5 etc..

Character constants

- A character constant is a constant which uses single quotation around characters.
- They have equivalent ASCII values.

  Example: 'a', 'l', 'm', 'F‘ etc..

String Constant

- A string constant is a sequence of characters enclosed in double quotes.
- It may contain letters, numbers, special characters or blank spaces. However it does not have equivalent ASCII values.

  Examples:- “Hello”, “2008”, “5+3” etc..

Preprocessor Directive

- C preprocessor is a collection of special statements, called directives, that are executed at the beginning of compilation process
- begin with the symbol # in column one and do not require a semicolon at the end
- Example
  - #define
  - #include

Symbolic Constant

- A symbolic constant is name that substitute for a sequence of character that cannot be changed.
- usually defined at the beginning of the program.

Example:

```
#define PI 3.141593
#define TRUE 1
#define FALSE 0
```

⟶ We can use symbol PI instead of constant term 3.141539 program.

Escape Sequence

- Certain nonprinting characters used in c are called escape sequences.
- An escape sequence always begins with a backslash and is followed by one or more special characters.

Example:	**Character**	**Escape Sequence**
	bell (alert)	\a
	backspace	\b
	horizontal tab	\t
	vertical tab	\v
	newline (line feed)	\n etc…..

C Tokens

- C tokens are the basic buildings blocks in C language which are constructed together to write a C program.
- Each and every smallest individual units in a C program are known as C tokens.

C tokens are of six types:

- Keywords **Example**: int, while etc…
- Identifiers **Example** : main, total etc…
- Constants **Example** : 10, 20 etc…
- Strings **Example** : “total”, “hello” etc…
- Special symbols **Example** : (), {} etc..
- Operators **Example** : +, /,-,* etc..

END. . .