Session 1 Introduction to C Programming Language

A COMPLETE C PROGRAMMING SERIES FROM BASICS TO ADVANCED

Facts about C language

- Developed at AT&T's Bell Labs of USA in 1972 by Dennis Richie.
- Is a middle-level language.

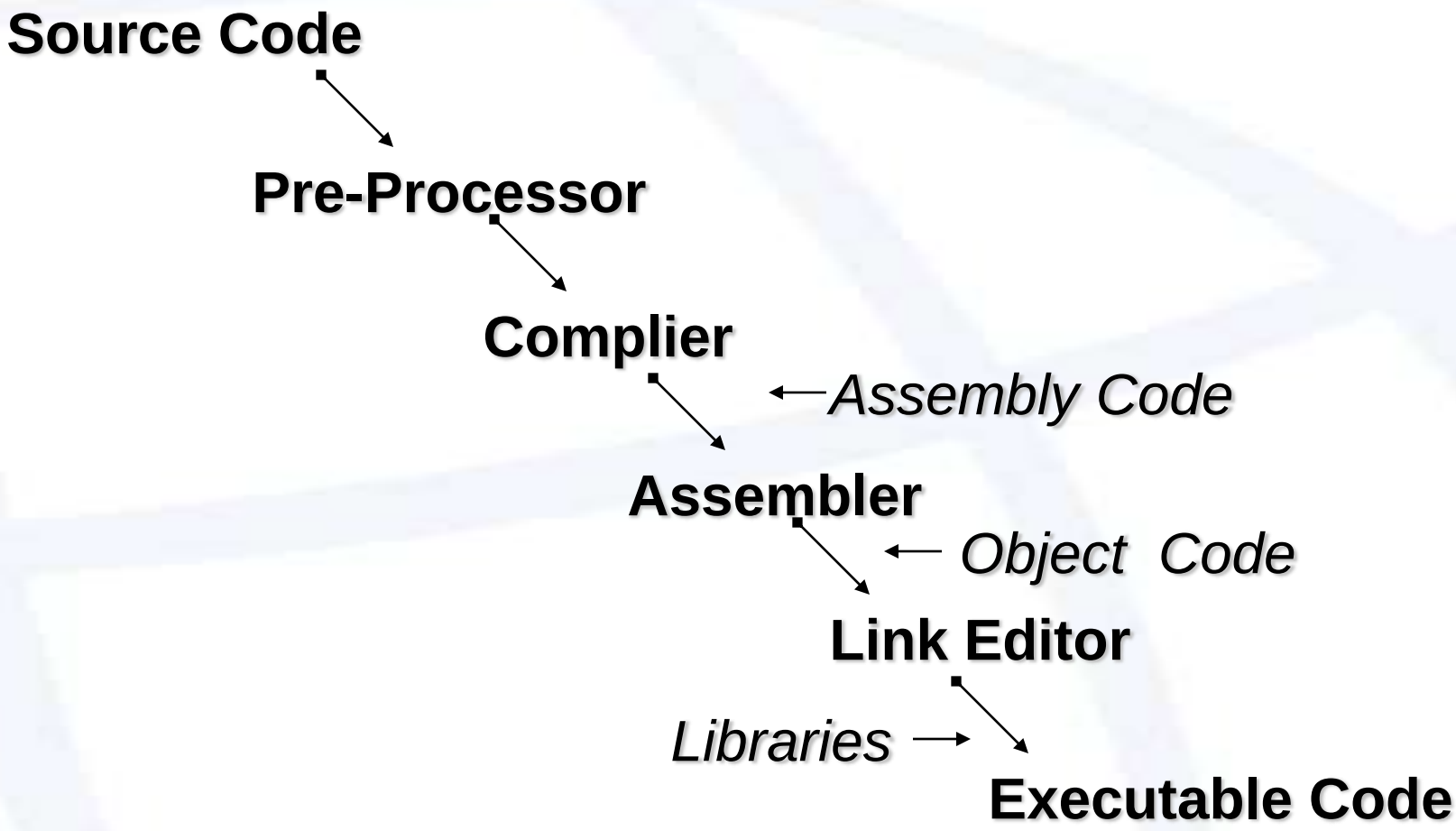
C-Program

- Statements in small case letters.
- Statements end with a semi-colon.
- Program execution starts from & ends with main().

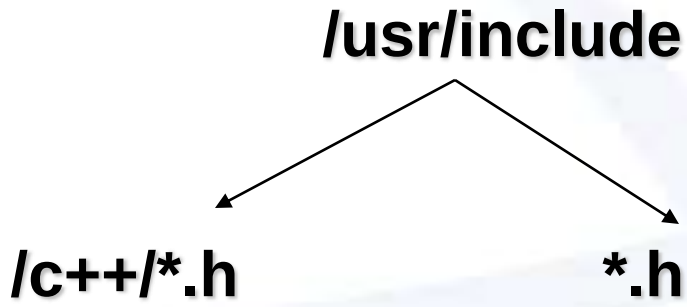
C-Program Structure

- Preprocessor Commands
- Type Definitions
- Function Prototype
- Variables and Functions

C Compilation Model



The Location of Header Files



Variables

- A Variable is a named location in the computer memory for holding some information whose contents can change characterized by a data type.
  - Values are assigned to a variable using an equals sign

    A = 5

    Assigns the value 5 to the variable A

Variable Names

- Variables have names which consist of letters, numbers and a few special characters.
- They must begin with a letter.
- The name has to be the combination of 1-31 characters. If Variable name is longer than 31 characters, first 31 characters are significant.
- No special symbol other than underscore is allowed in the variable name.

Data Types

Primary	Extended	Secondary
char	short	array
int	long	pointer
float		structure
double		union
void		enum

More about Data Types

Data Type	Size (in bytes)	Lower Bound	Upper Bound
char	1	-	-
unsigned char	1	0	255
short int	2	-32768	32767
unsigned short int	2	0	65535
long int	4	-2^{31}	$2^{31} -1$
float	4	$-3.2* 10^{98}$	$3.2* 10^{98}$
double	8	$-1.7* 10^{908}$	$1.7* 10^{908}$
Long double	10		10^{4096}

Integers

- Holds whole numbers (e.g. 1,2,3,4.....).
- Cannot hold fractional values (e.g. 3.2).
- Is limited to values between -32,768 and 32,767.
- Uses 2 bytes.
  - 1 Bit is used for sign (0 -> “+” and 1 -> “-”)
  - 15 Bits for Data

Strings

- Hold character data, such as names, addresses, quotes from a book, etc.
- Cannot be used for calculations.
- Are limited to between zero and approximately 64,000 letters.
- Use one byte per character.

Constants

Integer Constants

Range -- -32768 to 32767

Real Constants

Range -- -3.4e38 to 3.4e38

Character Constants

Range -- -128 to 127

Logical Constants

Any number except *zero* is considered as true.

String Constants

Is terminated by a null character.

Integer Constants Representation

Decimal – base 10

Octal – base 8

Hexadecimal – base 16

Comments

Non-executable lines of code

Placed for the purpose of documentation.

Keywords

- Predefined identifiers reserved for a specific job.
- Cannot be used as identifiers. Some C Compilers allows us to use them as identifiers.
- There are 32 keywords available in C.

  e.g. char, do, if etc.

Operators

Arithmetic

Relational

Logical

Increment & Decrement

Bitwise

Conditional

Mathematical Operators

- Mathematical Operators

  , /, +, -, %

- Increment and Decrement Operators

  + + → ***Increments the operand's value by 1.***

  - e.g. i = i + 1; can be written as **i ++;**

  - - → ***Decrements the operand's value by 1.***

  - e.g. i = i - 1; can be written as **i --;**

Notations of ++ and -- Operators

- Prefix Notation

  First Increments or Decrements and then continue with rest of operations. For example ***j = ++ i ;*** is actually implemented as

  i = i + 1; followed by j = i;

- Postfix Notation

  Completes rest of operations and then Increments or Decrements. For Example ***j = i ++;*** is actually implemented as

  j = i; followed by i = i + 1;

Relational Operators

There are six basic comparison operators:

<	Less Than
>	Greater Than
<=	Less Than or Equal To
>=	Greater Than or Equal To
=	Equal To
< >	Not Equal To

Logical Operators

- AND
  - Represented by the symbol ‘&&’.
  - Returns true if both the expressions are true.
- OR
  - Represented by the symbol ‘| |’.
  - Returns true if one of two expressions are true.
- NOT
  - Represented by the symbol ‘!’.
  - Reverses the result returned by the expression.

Expressions and Operators

- Expressions are built up using operators and operands. These concepts should be familiar to you from your math classes.
- Depending on the operators used and the purpose of the expression, they fall into three general groups:
  - Mathematical
  - Comparison
  - Logical
- These types may be mixed.

Type of Expressions

- Mathematical Expressions
  - Used to calculate values.
- Comparison Expressions
  - Compare two values and evaluate a relationship between them.
- Logical Expressions
  - Combine other expressions to form more sophisticated operations.

Examples of Expressions

x = 3 + 4 – 7 * 8 / 5 % 10

x = 3 / 2 + 3 * 8 / 3 – 3 + 1.5 / 3

x = (a != 10) && (b = 50)

x *= x + 4

z *= y = x * x

Cont….

Examples of Expressions Contd..

```
z = x++ + 10
x -=  --x  -  x--
z = ++x  &&  ++y  &&  ++z
z = ++x  &&  ++y  || ++z
x = ++i  *  ++i  *  ++i
```

Decision-making Constructs

- Relational & logical operators in the following constructs are used for decision-making :-
  - if - construct
  - if – else construct
  - switch – case construct

IF construct - Syntax

```
if (condition)
  {
  Statements to Execute if condition is true ;
}
```

if else construct - Syntax

```
if (condition)
{
    Statements to Execute if condition is true ;
 }
 else
 {
   Statements to Execute if condition is false ;
 }
```

Conditional Operator

- Expression1 ? Expression 2 : Expression 3
- If Expression1 returns true then Expression2 is evaluated else Expression3 is evaluated.

  e.g. x = (y > z ? y : z)

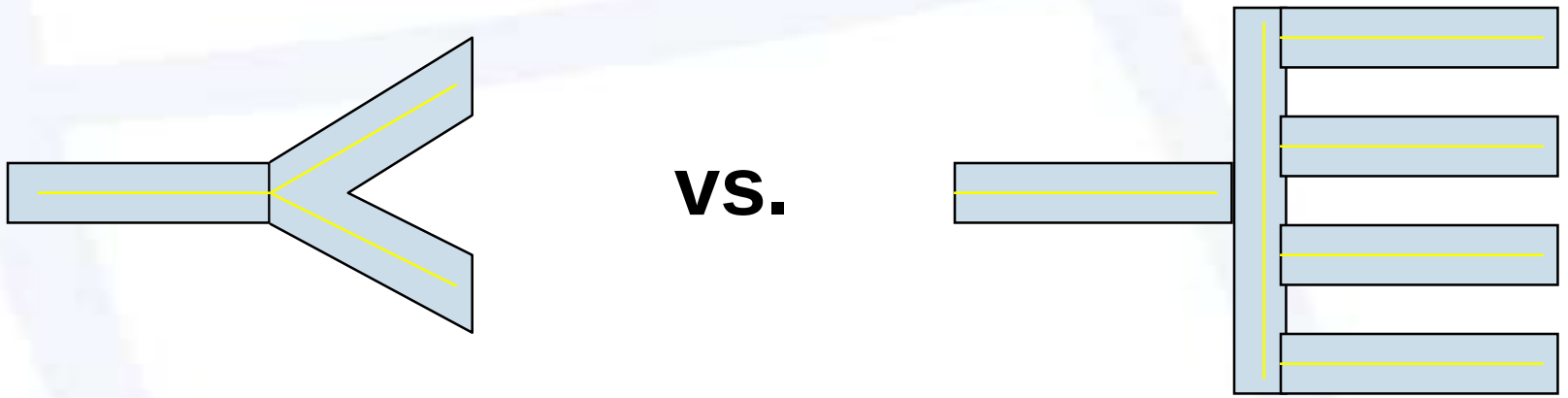
  If y > z then y is assigned to x else z is assigned to x.
- This operator can only replace the situation where a single statement exists in if and else both.

Nesting If Statements

- If statements can be placed inside one another.
- To get to the innermost execution area all the conditionals must be true... Which is precisely the same as And’ing them all together!

The switch case construct

- The switch case construct is used instead of the If - else construct when the decision is between many alternatives instead of two.



The *switch* case construct - Syntax

```
switch (test expression)
{
    case expression list1:
          First Statement block ;
          break;
    default:
        default Statement Block;
 }
```

The *break* statement in switch case

- The *break* statement in switch case construct causes an exit from the switch body and places the control to the first statement after the switch construct.
- If break statement is not incorporated after every case, the control moves to the next case statements and executes them.

The *default* statement in switch case

- The *default* statement in switch case construct gives a way to take action if the value of the switch variable does not match any of the case constants.

switch case construct - Example

```
switch (day)
 {
        case 1:
                printf ( " It is Monday." ) ;
                break;
        case 2:
                printf ( " It is Tuesday." ) ;
                break;
    default:
                printf ( " Wrong Input." ) ;
  }
```

Examples of Decision-making Constructs

```
a = 500
        if  (!a >= 400)
        printf (“True”);

if (a = b)
        printf (“True”);

if ( ! ( !x )  &&  x )
        printf (“True”);
```

```
k = -2, j = 4
        switch  (k /= j / k)
 {
        default :
                printf (“Same”);
        case 0 :
                printf (“0”);
        case 1 :
                printf (“1”);
 }
```

Cont….

End of Session 01