

Introduction to C

History of C:

- ✓ C is a structured/procedure oriented, high-level and machine independent programming language.
- ✓ The root of all modern languages is ALGOL, introduced in the early 1960.
- ✓ In 1967, Martin Richards developed a language called BCPL.
- ✓ In 1970, ken thompson created a language using many features of BCPL and called it simply B.
- ✓ In 1972, Dennis Ritchie developed a language from the best features of ALGOL,BCPL & B and called it simply C
- ✓ It was developed at AT&T bell labs.

Introduction to C

Characteristics / Features of C

- ✓ C is General purpose, structured programming language.
- ✓ C is highly portable i.e., it can be run in different OS.
- ✓ C is robust language, whose rich set of built in functions and operators can be used to any solve complex problems.
- ✓ C has ability to extend itself, we can continuously add our own functions to the existing system.
- ✓ C is well suited for writing both System S/W and Application S/W
- ✓ C program can be run on different OS with little or no modifications.
- ✓ C is also a middle level language i.e., it supports low level & middle level language features

Introduction to C

- ✓ C language allows reference to a memory location with the help of pointers, which holds address of memory location.
- ✓ C language allows dynamic memory allocation.
- ✓ C language allows to manipulate data at bit level.
- ✓ C programs are fast & efficient.
- ✓ C has rich set of operators.
- ✓ C are used to develop System programs like OS, Compiler & Assembler etc.
- ✓ C is a case sensitive programming language

Introduction to C

Structure of the C program

Documentation Section

Preprocessor Section

Definition Section

Global declaration section

```
main()
{
    declaration part;
    execution part;
}
```

Introduction to C

Documentation Section:

- ✓ it consist of set of comment lines used to specify the name of the program, the author etc.
- ✓ Comments are begin with `/*` and end with `*/`, these are not executable, the compiler is ignored any thing in between `/*`
`*/`
- ✓ Ex: `/* welcome to C world */`

Preprocessor Section:

- ✓ One major part of the C program is preprocessor.
- ✓ The preprocessor directives are commands that give instructions to C preprocessor.
- ✓ Whose job is modify the text of the C program before it compiled.

Introduction to C

- ✓ A preprocessor begin with #.
- ✓ Two most common preprocessor directives are #include & #define.
- ✓ Every C compiler contains collection of predefined function & symbols.
- ✓ Pre defined functions & symbols are organized in the form of header files whose name ends with .h.
- ✓ For Ex: stdio.h, conio.h etc.
- ✓ The #include directive causes the preprocessor to insert definitions from a standard header file.
- ✓ For Ex: #include<stdio.h>
- ✓ Stdio.h having libaray functions like printf(), scanf() etc

Introduction to C

Definition Section

- ✓ The definition section defines all symbolic constants.
- ✓ For Ex: `#define PI 3.142`.

Global Declaration:

- ✓ the variables that are used in more than one function throughout the program are called global variables and are declared out side of all the functions.

main() function:

- ✓ Every C program must have one `main()` function, which specify the starting of C program.
- ✓ Every C program execution begin from `main()` only.
- ✓ `main()` function is entry point of the program.

Introduction to C

- ✓ C program allows any number other functions, which are called **user defined functions**.
- ✓ Every C function contains two parts.
 - 1-> **declaration part:** this part is used to declare all the variables that are used in the executable part of the program and called **local variables**.
 - 2-> **executable part:** it contains at least one valid C statement.
- ✓ Every function execution begins with opening brace { and end with closing brace }.
- ✓ Note: the closing brace(}) of the main function indicate end of the program.

Note: C program is a collection of functions.

Introduction to C

Sample C program

```
/* this is a simple C program */ documentation
#include<stdio.h>      pre-processor section
void main()
{
    printf("Welcome to C");
}
```

- ✓ Every C program stored on a disk in the form of **file**.
- ✓ File : file is named collection of data & instructions.
- ✓ Every file name contains two parts which are separated by dot(.)
- ✓ First part is name of the file and second part is extension.
- ✓ For Ex: add.c **add** is name of the file & **.c** is extension.
- ✓ Extension defines type of the file.

Introduction to C

- ✓ Name defines purpose of the file.
- ✓ Dictionary : it is a collection of files.
- ✓ All the statements in the program ends with a semicolon(;) except conditional & control statements.

Programming Rules

- ✓ All the statements in C program should be written in lower case letters. Upper case letters are only used for symbolic constants.
- ✓ The program statements can be write anywhere between two braces({, }) .
- ✓ The programmer can also write one or more statements in one line separating them with a semicolon.
- ✓ C is a free form language.
- ✓ C is a case sensitive language

Introduction to C

Executing a 'C' program

- ✓ Execution is the process of running the program,
- ✓ To execute a 'C' program, we need to perform the following steps.

1)Creating the program: creating the program means entering or editing the program by using standard 'C' editor and save the program with .c as an extension.

- ✓ For Ex: sample.c.
- ✓ The popular C editors are turbo c, borland c, ANSI C etc.

2)Compiling the program: this is the process of converting the high level language to machine level language.

- ✓ The above process is performed ,only when the program is syntactically & semantically correct .

Introduction to C

- ✓ The mistakes in a program are called errors.
- ✓ In C program errors are broadly divide into two types
- ✓ 1)syntax errors 2)logical errors
- ✓ Syntax: it is a set of rules & regulations of programming language.
(Or) grammar of the programming language

Syntax Error:

- ✓ the grammatical errors in the program are called syntax errors.
- ✓ Syntax errors are easy to correct, because of C editor provide brief description about the error along with line number.
- ✓ During the compilation, C compiler scan the entire program to detect syntax errors.
- ✓ Error free programs only compiled.
- ✓ Once the program is successfully compiled, it will generate three different files.

Introduction to C

- ✓ These are **.bak, .obj, .exe**.
- ✓ .bak are backup file, which are used for recover the source code.
- ✓ .obj are object file, it is collection of machine instructions that is output of compiler.
- ✓ .exe are the executable files, which are used for executing the program.

Note: C programs are compiled by using functional key F9.

Linking the program with system library.

- ✓ The linker is a systems program ,it combines user object file and library object files and also perform cross references.
- ✓ The linker output is .exe file and store on the disk.

Introduction to C

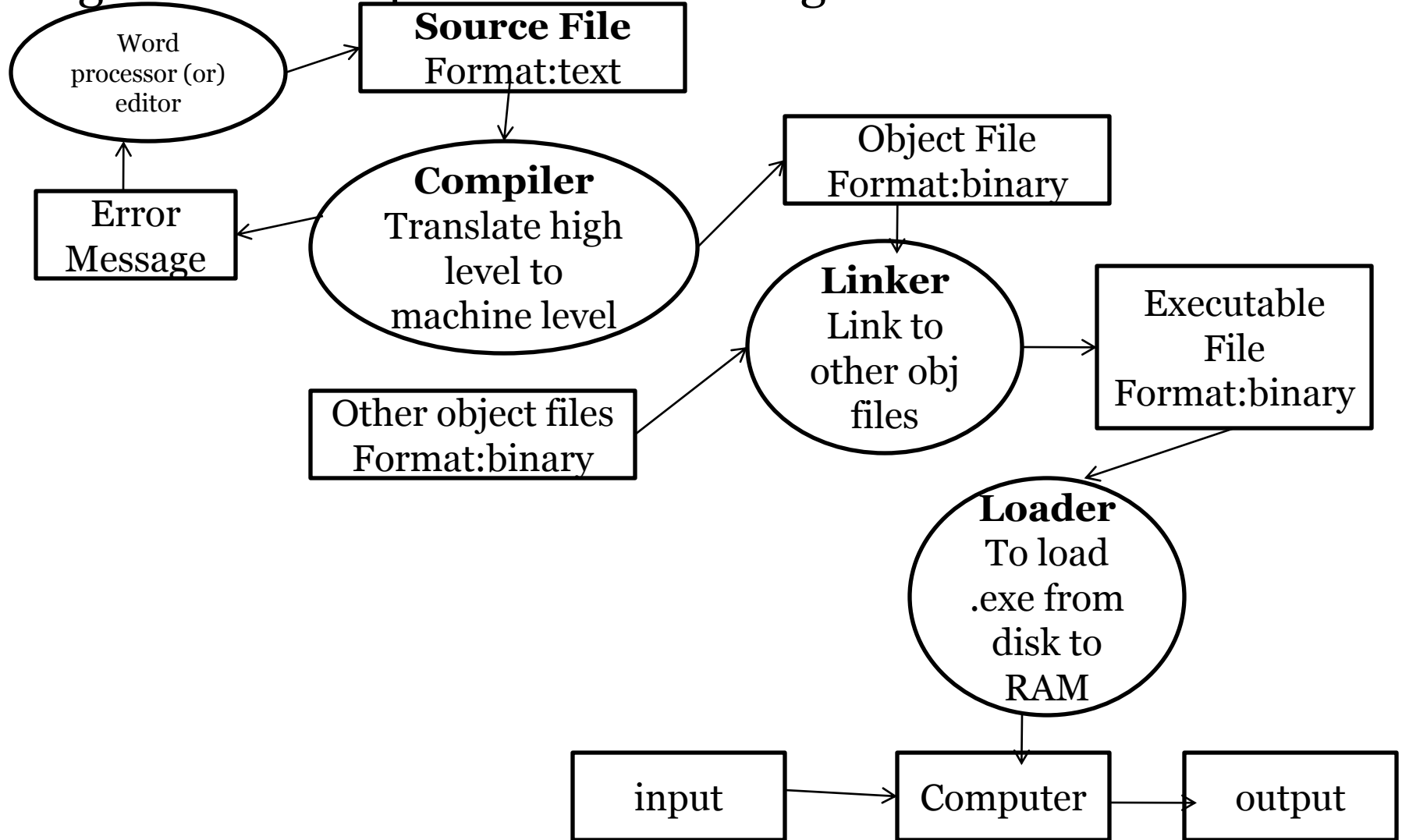
- ✓ .exe is executable file: it is collection of machine instructions and are ready to execute under CPU.

Executing the program

- ✓ Loader is a systems program, it loads .exe file from disk to RAM and informs to the CPU beginning of the execution.
- ✓ Semantics: semantics are nothing but meaning of the identifier.
- ✓ C program execution always begin from main() function
- ✓ ***Note: C programs are executed by using CTRL + F9.***

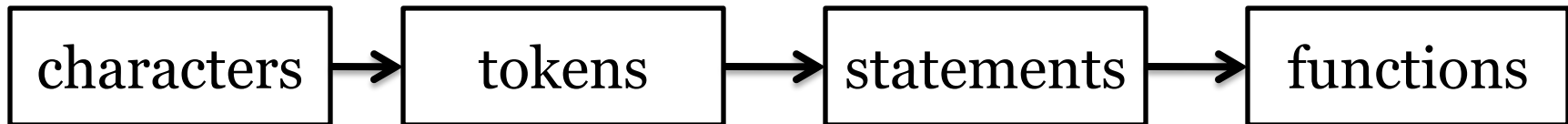
Introduction to C

Diagrammatic Representation C Program Execution



Basics of C

Learning Stages of C Language:



C character Set:

Alphabet : A – Z, a – z

digits : 0 – 9

special Characters: + = _ -) (* & ^ % \$ # @ ! ~ ` “ ‘ ; , . / ?

white spaces/escape sequence : blank space, tab, new line
etc

- ✓ C uses ASCII character Set.
- ✓ There are 128 ASCII symbols
- ✓ Each character represents by one byte.

Basics of C

- ✓ Each ASCII symbol having unique ASCII value.

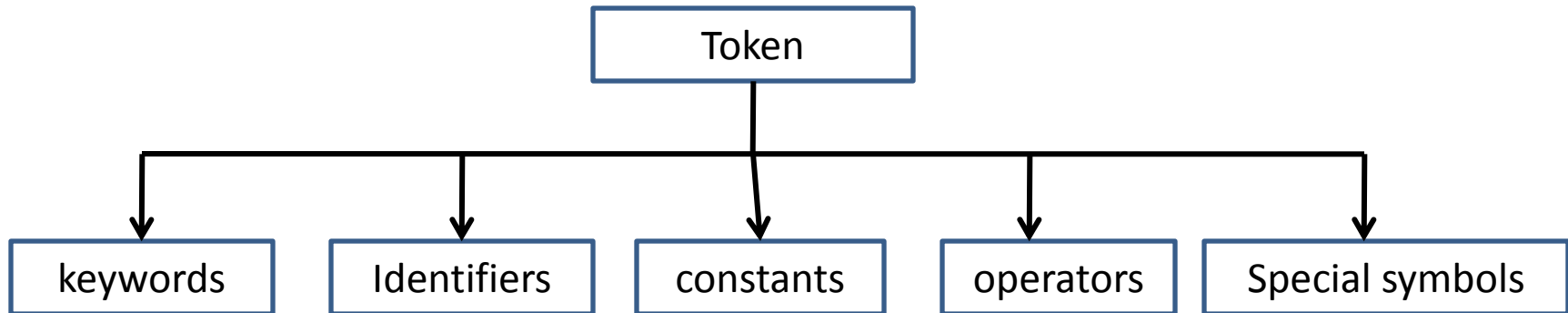
For Ex: 0 49 * 43
 1 50) 42
 enter 13
 A 66
 a 98

Tokens:

- ✓ Smallest individual units in C program are known as 'C' Tokens.

Basics of C

- ✓ C has 5 types of tokens



Key words

- ✓ There are certain reserved words called token, that have standard & predefined meaning in C language.
- ✓ Whose meaning can't be changed.
- ✓ These are building blocks for C program statements.
- ✓ C having 32 keywords.
- ✓ For ex: int, for, while,if,else,struct,union,return etc.
- ✓ All keywords must written in lower case.

Basics of C

Identifiers:

- ✓ Identifiers are names given to various program elements, such as variables, functions and arrays etc.

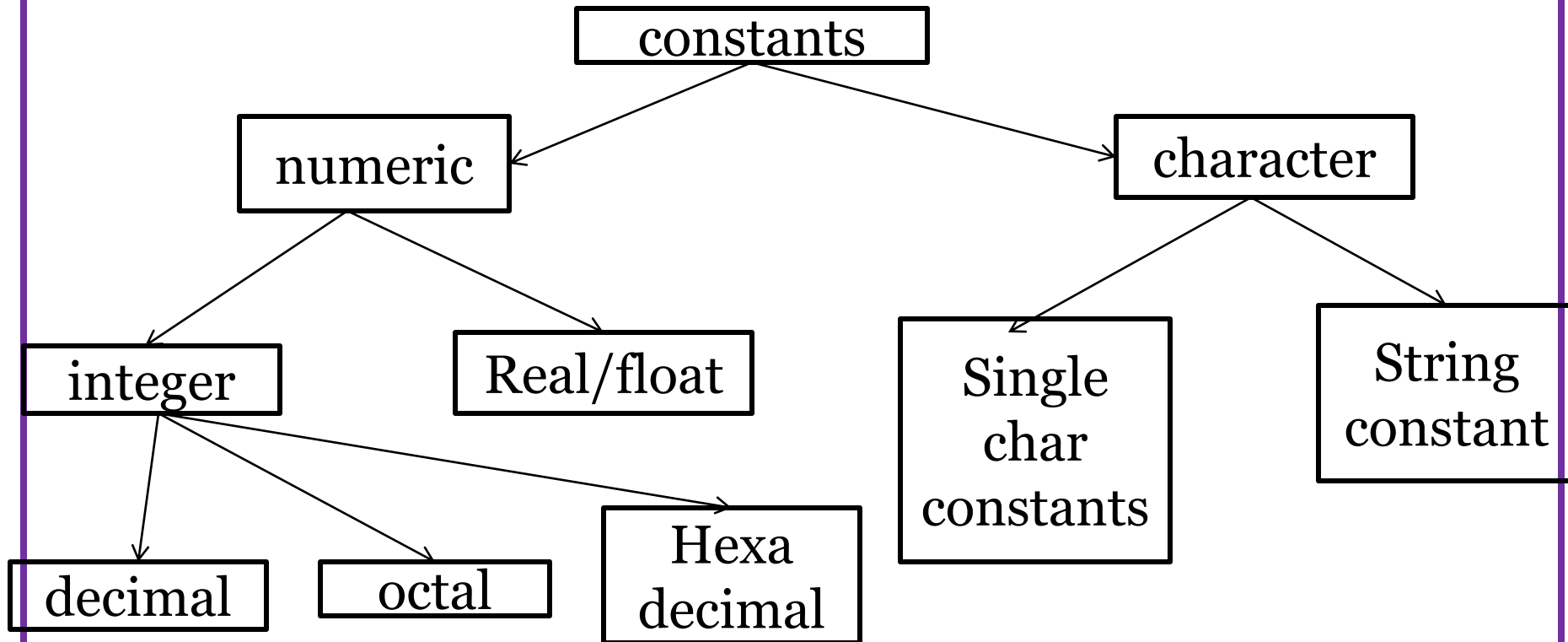
Rules for naming an identifier

1. Identifier consists of letters, digits & special symbol.
2. It allows one and only special character i.e. underscore (_).
3. The first character can't be digit.
4. Upper case are different to lower case.
5. An identifier can be any length, preferred size is 31 characters
6. The identifier cannot be a keyword.
7. Does not allow space between the words in an identifier.

Basics of C

Constants:

- ✓ Constants refer to fixed values that do not change during execution of program
- ✓ Constants are broadly divided into following sub types



Basics of C

Integer Constants:

- ✓ These refers to integers/whole numbers consisting of sequence of digits.
- ✓ There are three types of integer constants
- ✓ 1. decimal 2.octal 3.hexa decimal

Rules for defining integer constants.

- 1.An integer constant must have at least one digit.
- 2.It should not contain any special symbols & white spaces.
- 3.It should not contain any decimal point or exponent.
- 4.The integer value can't exceed the range allowed the particular machine.

Basics of C

Decimal Constant

- ✓ Decimal integer constant range from 0 to 9 & preceded by optional sign.
- ✓ Decimal integer constant first digit must not be zero.

valid

+129

-95

invalid

1,29

1 54

Octal Constants:

- ✓ Octal integer constants are in the range from 0 to 7.
- ✓ Sign is optional.

Basics of C

- ✓ Every octal integer constant preceded with 0(zero).

valid

invalid

037

081

0

541

036

01,35

Hexadecimal constants

- ✓ Hexadecimal integer constants is a combination of digits from 0 to 9 & alphabets A to F represents from 10 to 15.
- ✓ Every hexadecimal number precede with 0X.

valid

invalid

0XA5

075

0XABC

0XAGE

0X9FA

0X7,AF

Basics of C

Real/Float Constants

- ✓ These constants refer to the numbers containing fractional parts.
- ✓ These are also known as floating-point constants.
- ✓ Two ways of representing real constants
- ✓ 1. decimal form 2. exponential form/scientific form.
- ✓ In decimal form, decimal & fractional part are separated by .(dot)

Valid

0.056

-6.453

invalid

0.78.78

-89.34 90

Basics of C

- ✓ Exponential form consists of two parts mantissa & exponent.
- ✓ Exponent & mantissa are separated by e or E.
- ✓ Exponent define to shift decimal point to the right if exponent is positive or to the left if exponent is negative.
- ✓ If decimal point is not included with in the number assumed to be positioned to the right of last digit.

Rules for constructing real/float constants

- 1.Mantissa & Exponent can be either positive or negative.
- 2.Special symbols are not allowed except .(dot)
- 3.Exponent must be an integer.
- 4.Exponent & Mantissa must have at least one digit each.

Basics of C

For Ex: 4×10^4 can be represented as floating point constant as

40000.

4E4/4E+4

400E2

For Ex: 3.04×10^{-5} can be represented as floating point constant as

3.04E-5

30.4 E -6

0.00304E-2

Invalid numbers

E+10

3.45e8.9

Basics of C

Character Constants

✓ These refers to single character enclosed with in single quote marks.

✓ For Ex: valid invalid

✓ 'A' "A"

✓ '2' 'abc'

✓ '\$' "\$"

String Constants

✓ These refers to group of characters enclosed with in double quote.

✓ For EX: valid → "Hello", "A", "2", "+"

invalid → 'hello', 'A', '2', '+'

Basics of C

Variables:

- ✓ Def: the variable is an identifier, which holds data during program execution.
- ✓ Variables are hold different values during execution of the program.
- ✓ The rules to define a variable are similar to an identifier.
- ✓ The syntax of declaring a variable are
- ✓ **data type name of the variable;**
- ✓ **Data type:** the term data type refers kind of data or type of data involved during execution of program.
- ✓ Name of the variable is a group of characters.
- ✓ Variable are broadly divided in to two categories
- ✓ 1.local variables 2.global variables

Basics of C

- ✓ Local variables are variables which are declared inside functions.
- ✓ Global variables are variables which are declared outside of all the functions.

Valid

account_no

sum

_a

list_of_words

a78count

a67

Invalid

account no

sum&

-a

a123 sum

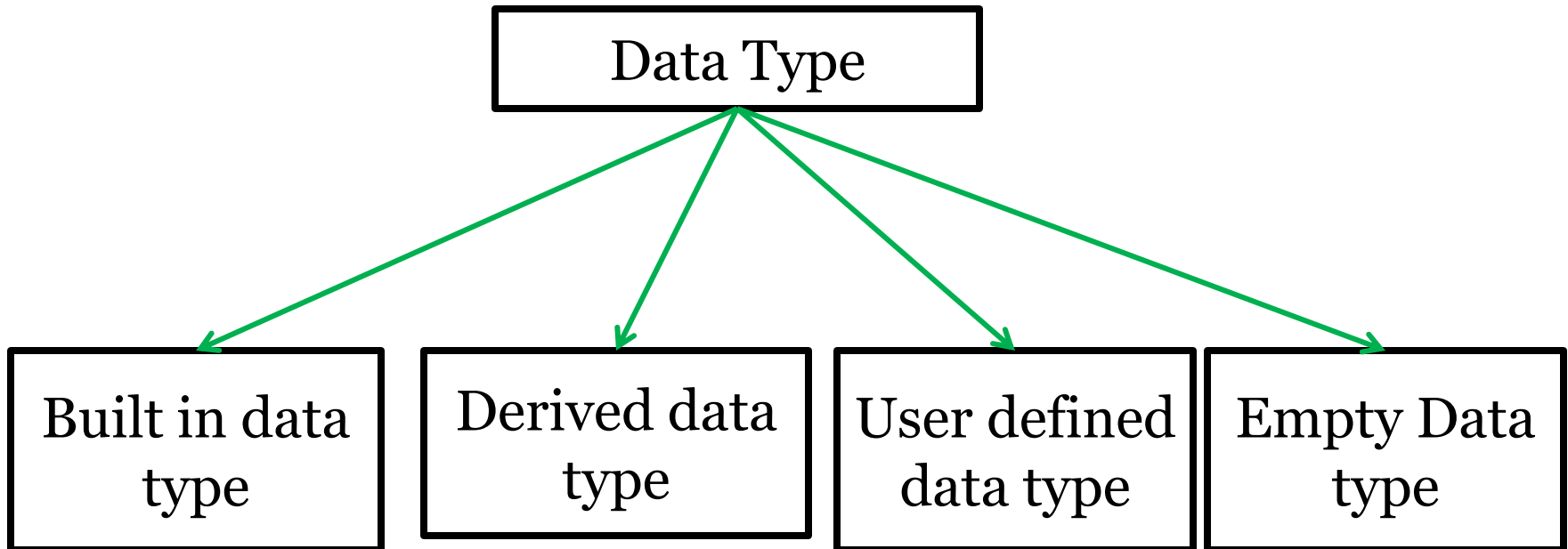
a+78_count

67a

Data Types in C

Def: the term data type refers kind of data or type of data involved during computation.

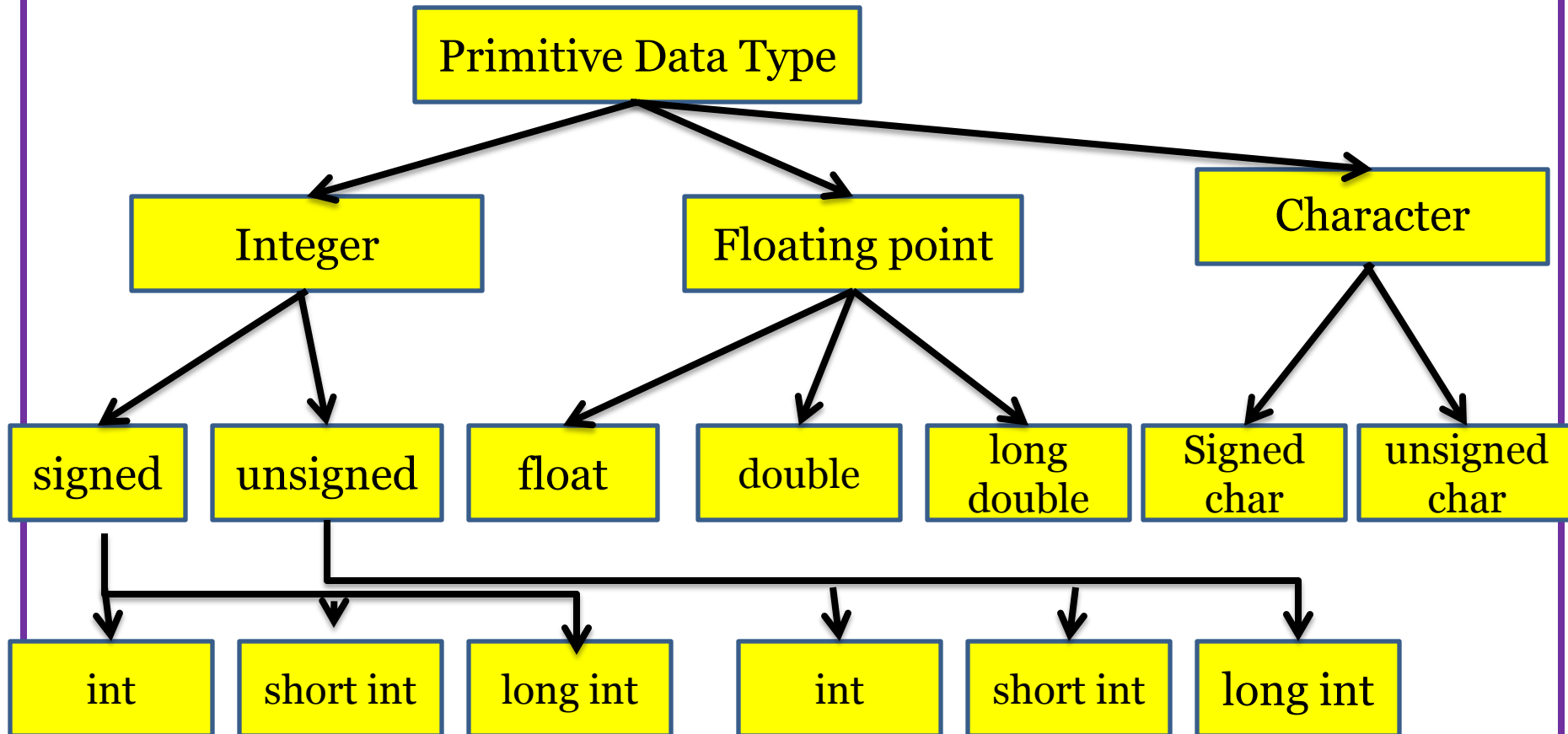
- ✓ Each variable or data item in C program associates with one of the data type.
- ✓ C data types are broadly divided in to 4 types.



Data Types in C

Built-In data type/primary data type/primitive data types.

- ✓ Built-in data types are further divided into following categories.
- ✓ The division is performed based on the type of data and amount of memory required for specific data.



Data Types in C

The Size and Range of primitive data types

Data Type	Byte	Range	Format Specifier
char or signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
int or signed int	2	-32768 to 32767	%d
unsigned int	2	0 to 65535	%d or %u
short int or signed short int	1	-128 to 127	%d
unsigned short int	1	0 to 255	%d or %u
long int or signed long int	4	-2,147,483,648 to 2,147,483,648	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
float	4	-3.4E-38 to 3.4E38	%f or %g or %e
Double	8	-1.7E-308 to 1.7E308	%lf
long double	10	-1.7E-4932 to 1.7E4932	%lf
Octal Decimal number			%o
Hex Decimal number			%x

Data Types in C

Examples for variable declaration

- ✓ `int sum;` or `signed int sum;`
- ✓ `int a,b,c;` or `signed int a,b,c;`
- ✓ `unsigned int regno;`
- ✓ `float interest;`
- ✓ `double amount;`
- ✓ `long int basic;`
- ✓ `char grade;`

Operators in C

Expression

- ✓ Def: An expression is a collection of operands and operators.
- ✓ The operand is either a variable or literal.
- ✓ The operators are broadly divided in to following categories

Name	operator	precedence	associativity
Unary	-	1	R-> L
increment & decrement	++, --	2	R-> L
Arithmetic	*, /, % +, -	3	L -> R
		4	
Relational	<, >, <=, >=, ==, !=	5	L -> R
Bitwise	& (AND), (OR) ^(XOR), <<(shift left), >>(shift right), ~(ones complement)	6	L -> R

Bit Wise Operators

- Bitwise operator works on bits and perform bit by bit operation.
- Assume if A = 60; and B = 13; Now in binary format they will be as follows:

- A = 0011 1100

- B = 0000 1101

- -----

- A&B = 0000 1100

- A|B = 0011 1101

- A^B = 0011 0001

- ~A = 1100 0011

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
	Binary OR Operator copies a bit if it exists in either operand.	(A B) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -60 which is 1100 0011
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

Operators in C

Continued...

Logical	&&(AND) (OR), !(NOT)	7 8	L -> R
Conditional	? :	9	R -> L
Assignment	=	10	R -> L

Special Operators

comma	,		L -> R
Function call	()		L -> R
Structure operator	->		L -> R
Size of	sizeof		L -> R
Address of operator	&		R -> L
Value at address operator	*		R -> L

Operators in C

Continued ...

Compound Assignment	+=,/=, -=, %=		R -> L
Array Exp	[]		L -> R

Arithmetic Operators

- ✓ For arithmetic operations, operands either numeric or character.
- ✓ Modulo (%) division operator applied only for integers.
- ✓ An expression with arithmetic operators are called **arithmetic expression**.
- ✓ Arithmetic Expressions are of three types
1.Integer Arithmetic 2.floating arithmetic 3.mixed arithmetic

Operators in C

- ✓ Integer arithmetic always yields integer value
- ✓ In modulo division the sign of the result will be same as the first operand.

Relational Operators

- ✓ They are used to compare two expressions/values.
- ✓ Its syntax are `exp1 relaop exp2`
- ✓ Result of the relational expression is either 0 or 1.
- ✓ Relational Expressions are used in decision statements.
- ✓ Relational operators are `>`, `<`, `>=`, `<=`, `==`, `!=`.
- ✓ An Expression with relational operators are called ***Relational Expression***.

Operators in C

Logical Operators:

- ✓ These are used to combine two or more expressions.
- ✓ There are 3 logical operators.

AND(&&)

- ✓ the result of the logical AND expression will be true only when both the expressions are true.
- ✓ The result of the expression will be either 0 or 1.
- ✓ Its syntax are **exp1 && exp2**

OR(||)

- ✓ The result of the OR expression are false only when both the expressions are false.
- ✓ Its syntax are **exp1 || exp2**

Operators in C

Not(!)

- ✓ The result of the expression will be true if the expression is false and vice versa.

Increment and Decrement

- ✓ These operators are represented by ++, --.
- ✓ ++ increment by 1.
- ✓ -- decrement by 1.
- ✓ These are unary operators.
- ✓ These are take the following form.

operator	meaning
a++	post increment
++a	pre increment
a--	post decrement
--a	pre decrement.

Operators in C

Conditional Operator(? :)

- ✓ It is also called ternary operator.
- ✓ Its syntax are **variable-name = exp1 ? exp2 : exp 3;**
- ✓ The value of the exp1 evaluated first, if it is true, exp2 evaluated otherwise exp3 evaluated.
- ✓ It is an alternative of ***if-else*** statement.

Evaluation of Expression.

- ✓ An Expression evaluation depends on associativity and precedence of operators.
- ✓ An expression is evaluated when it is terminated by semicolon(;).
- ✓ For Ex: $y = 2 + 5 * 3 - 6 / 3;$

Expressions in C

- ✓ $2 + 15 - 6/3$
- ✓ $2 + 15 - 2$
- ✓ $17 - 2$
- ✓ $15.$

Note: to change the order of evaluation by inserting parenthesis's in to the expression.

- ✓ The number of left & right parenthesis's must be the same, otherwise expression is invalid.
- ✓ If more than one level of parenthesis's exists, innermost evaluate first and so on.

For Ex: $y = (2 + 5) * (13 - 6) / 3;$

- 1) $7 * (13 - 6) / 3$
- 2) $7 * 7 / 3$
- 3) $49 / 3$
- 4) 16

Expressions in C

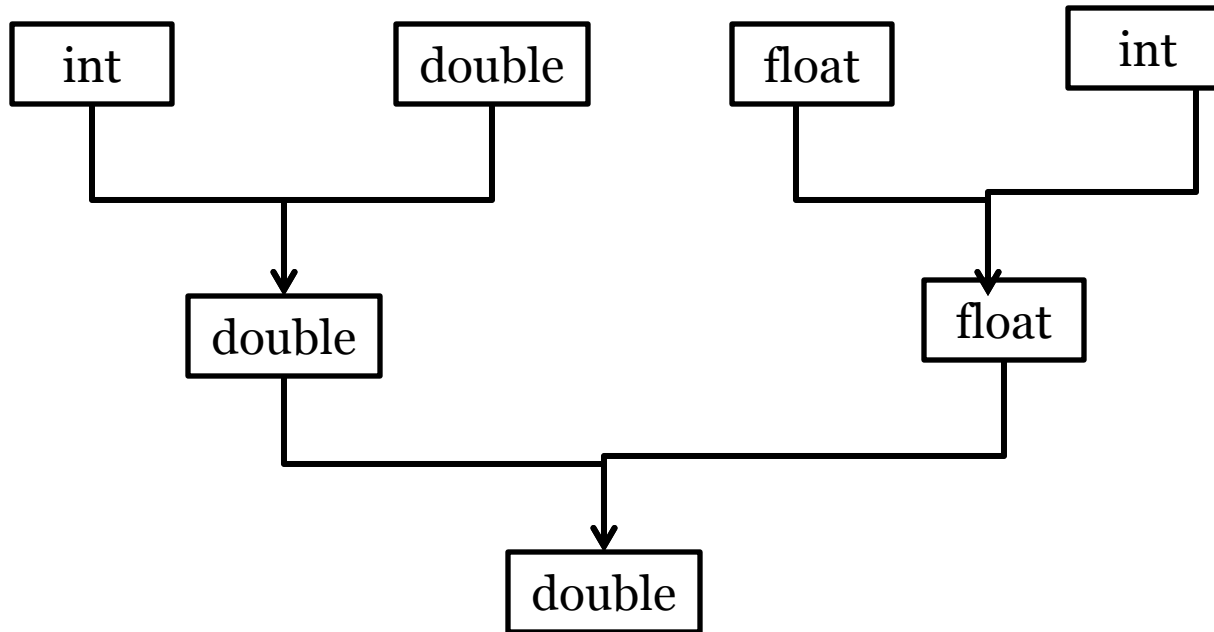
Evaluation of Expression in Mixed Mode Arithmetic:

- ✓ When an expression consists of different types of variables, C compiler follows rules of **type conversion**.
- ✓ There are two type of conversion
- ✓ 1. implicit conversion/automatic conversion/integral promotion/implicit casting.
- ✓ 2.explicit conversion / type conversion/type casting.
- ✓ ***Implicit Conversion:*** when the data type of smaller size is converted into higher size.
- ✓ It will be done by compiler itself.
- ✓ Implicit conversion is also called widening, because of smaller values are promoted to higher data types.

Expressions in C

For Ex: int i, double d,e and float f

$e = i * d + f / i;$



Note: Result of the Expression in double Data-Type.

Expressions in C

- ✓ Apart from simple assignment operator, C provides compound assignment operators to assign a value to a variable after performing a specific operation.
- ✓ Some of the compound assignment operators and their meanings are given below

Operator	Example	Meaning
<code>+=</code>	<code>X +=Y</code>	<code>X = X + Y</code>
<code>-=</code>	<code>X -=Y</code>	<code>X = X - Y</code>
<code>*=</code>	<code>X *=Y</code>	<code>X = X * Y</code>
<code>/=</code>	<code>X /=Y</code>	<code>X = X / Y</code>
<code>%=</code>	<code>X %=Y</code>	<code>X = X % Y</code>

Expressions in C

Nested (OR) Multiple assignments.

- ✓ Using this feature we can assign a single value or expression to multiple variables.

For Ex:

```
int a,b,c,x,y,z;
```

```
a = b = c = 20;
```

```
x = y = z = (a+b+c);
```

Variable Declaration;

- ✓ Variable Declaration tells the compiler what the variable name and type of data that variable is going to hold.
- ✓ Default value hold the variable after declaration is called **garbage value/unknown value.**

For Ex: int a;

Variables in C

Variable definition/ Initialization.

- ✓ To assign some value to the variable at the time declaration is called variable definition.
- ✓ For Ex: `int i = 0;`

Note: In C programming all the variables are declared/ defined before the first executable statement.

Empty Data Type

- ✓ It is also known as **void** data type.
- ✓ It indicates no other data type has been used with the given identifier.
- ✓ It is used for function return type, which are not interested to return any value.
- ✓ It is also used for creating **generic pointers**.

Escape Sequence in C

Escape Sequence (or) Backslash Characters.

- ✓ each character having its own meaning and they contain two characters and first character must be \ (slash).
- ✓ These set of characters are also called as non-graphic characters.
- ✓ These characters are invisible and cannot be displayed directly.

Escape Sequence in C

✓ The following table shows list of escape sequence

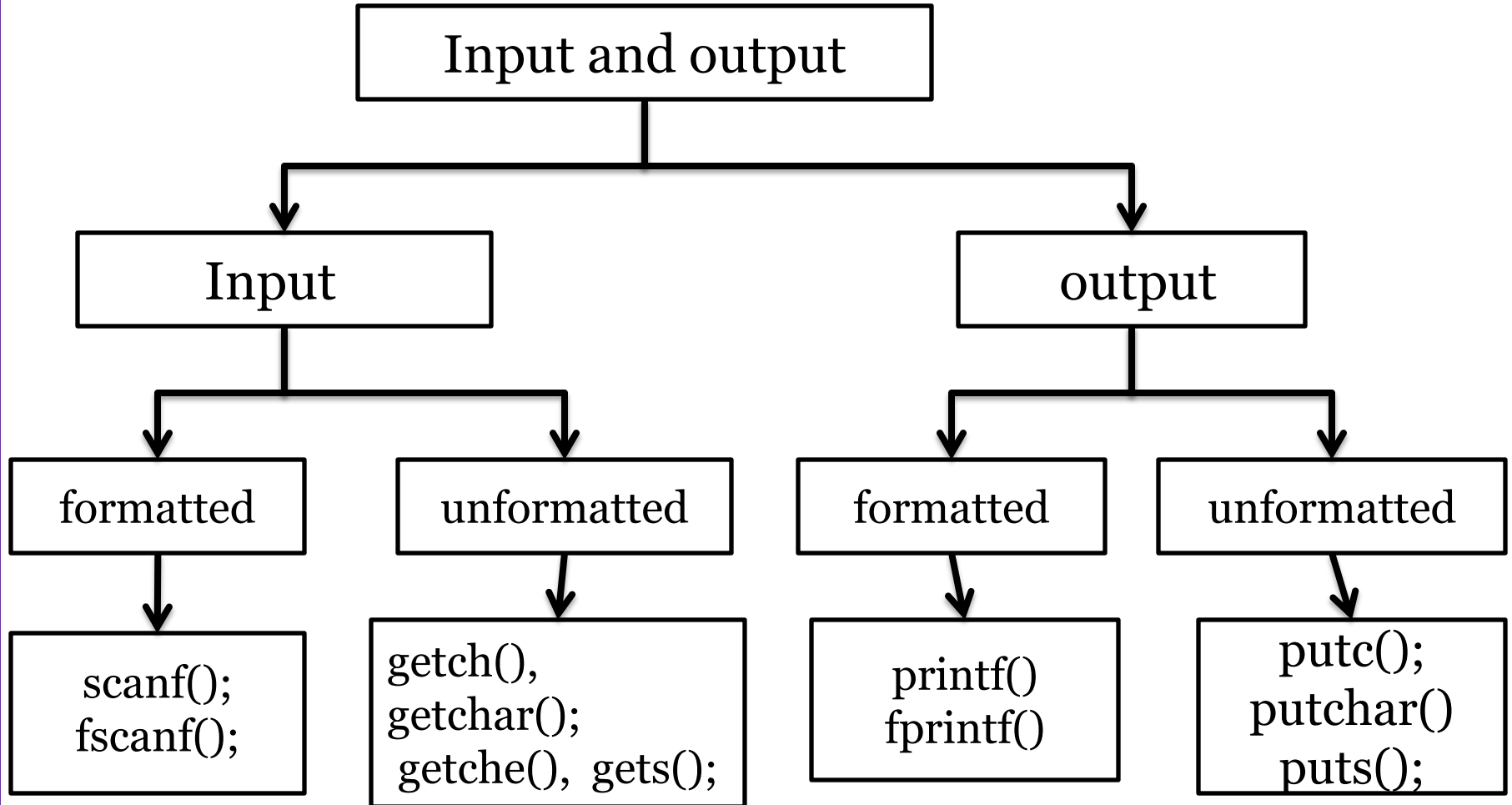
Character	Escape sequence	ASCII value	Meaning
Bell	\a	007	Beep Sound
Backspace	\b	008	Moves previous position
New Line	\n	010	Moves next line
Form feed	\f	012	Moves initial position of next page
Carriage Return	\r	013	Moves beginning of the line
Quotation mark	\"	034	Present double quote
Back slash	\\	092	Present back slack
Null	\0	000	It indicate end of string.

Input / Output in C

- ✓ We know that input, process and output are the essential features of computer program.
- ✓ **Input:** input is the process of accept data from standard input device(keyboard etc).
- ✓ **Output:** output is the process of display information on the standard output device(VDU/monitor etc).
- ✓ There are two methods for providing data to the program.
 1. Assigning the data to the variables in a program.
 2. By using the I/O functions.
- ✓ I/O operations performed using predefined library functions.
- ✓ These are classified into two types

Input / Output in C

- ✓ Formatted I/O and unformatted I/O
- ✓ Broad classification of I/O functions are



Input / Output in C

Formatted Output

- ✓ Formatted output refers to display information in a particular format.
- ✓ `printf()` is used to display information on standard output device.
- ✓ This function display any combination of data.
- ✓ Its general syntax are

printf("control string", list of variables);

- ✓ ***Control String:***
- ✓ It enclosed with in double quota.
- ✓ It specify type of data to be displayed on the output device.
- ✓ Control string consists of format specifier, it precedes with %.

Input / Output in C

- ✓ It also consists of escape sequence and string that to be displayed on the output device.
- ✓ By default output of the printf() function is left justified.
- ✓ Filed with in format specifier makes out is right justified.

For Ex: `int a = 450,b = 3478,c = 12;`

```
printf("\n %d \n %d \n %d",a,b,c);
```

Out Put is

450

3478

12

- ✓ `printf("\n %4d \n %4d \n %4d",a,b,c);`

Input / Output in C

450

3489

12

```
float g = 3456.8935;
```

```
printf(“%4.2f”,g);
```

Out put is 3456.89

- ✓ - Indicates the output is left justified.
- ✓ + indicates the output is displayed with sign.

For Ex:

```
printf(“%-10d”,k);
```

```
printf(“%+d”,k);
```


Input / Output in C

Rules for writing printf() function

1. Place the appropriate headings in the output.
2. The variable must be separated by comma and need not be preceded with ampersand (&).
3. The control string and variables must match their order.
4. Provide the blank space between the numbers for better clarity.
5. Print special messages wherever required in output.

Input / Output in C

Formatted Input

- ✓ Formatted input refers to read data in a particular format.
- ✓ scanf() is used to read data from the standard input device.
- ✓ This function read any combination of data.
- ✓ The syntax of scanf() function are

scanf("control string ",list of variables/address list/input list);

Control string

1. It enclosed with in double quote.
2. It specifies type of data that have to be read from input device.
3. Control string consists of format Specifier, it preceded with a % sign.
4. The control string and variables must mach their order.

Input / Output in C

List of variables

- ✓ List of variables separated by ,(comma).
- ✓ Each variable preceded by an ampersand(&).
- ✓ It specifies the address of variable.
- ✓ Ampersand(&) is also called ***address of operator***.

For Ex: `scanf(" %d %f %c",&a,&b,&c);`

Suppose we enter 10 5.5 g

10 is assigned to a

5.5 is assigned to b

g is assigned to c.

Input / Output in C

✓ We can also specify field width in the format specifier.

For Ex: `scanf("%3d %2d",&a,&b);`

Suppose we enter 500 20

500 is assigned to a.

20 is assigned to b.

Note :it is not always advisable to use field width in formatter specifier, it may assign wrong values to variables.

For Ex: `scanf("%2d %3d",&a,&b);`

Suppose we enter 5004 10

50 is assigned to a

04 is assigned to b

10 is ignored.

Input / Output in C

- ✓ * Is used to suppress the input.
- ✓ For Ex: `scanf("%d %d %*d %*d %d",&a,&b,&c);`
- ✓ Our input is 10 20 30 40 50.
- ✓ 10 is assigned to a
- ✓ 20 is assigned to b
- ✓ 30 is suppressed
- ✓ 40 is suppressed
- ✓ 50 is assigned to c.

Control Statements

- ✓ Control statements are building blocks of C programming.
- ✓ Control statements determine the flow of execution.
- ✓ There are three types of control statements
 - 1) Sequential Statements.
 - 2) Conditional Control Statements.
 - 3) Loop Control Statements.

Sequential Statements

- ✓ The sequential statements are executed one after another from top to bottom.
- ✓ In this section every statement is executed exactly once.

Control Statements

Conditional Control /Selection/Decision Statements.

- ✓ The execution of a statement(s) is depends on result of the condition/expression.
- ✓ The selection statements create multiple paths in program.
- ✓ There are five different selection statements.

Control Statements

- 1) Simple if
- 2) if – else
- 3) Nested if
- 4) if – else – if ladder
- 5) switch - case

Simple if

- ✓ The if statement is a decision making statement.
- ✓ ***if*** is a keyword.
- ✓ It is used to control the flow of execution (or) it is used select set of statement(s) to be executed.
- ✓ It is always used in conjunction with expression.
- ✓ The general syntax of simple if are

Control Statements

if (expression)

st;

} if block

next-statement;

if (expression)

{

st1;

st2;

stn;

}

next-statement;

} if block

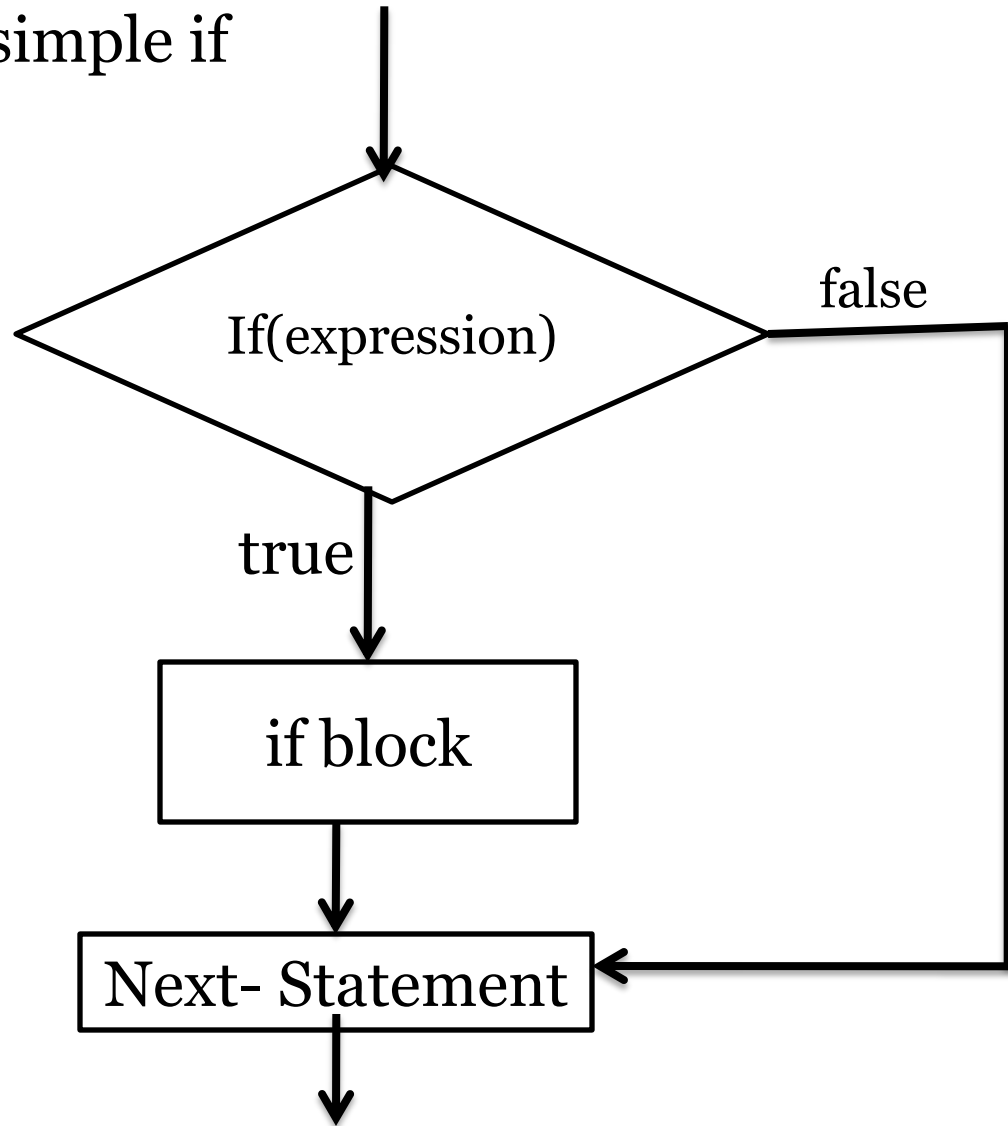
Control Statements

Properties of if Statement

- ✓ First evaluate the expression.
- ✓ If expression is true(1) , if block is executed and followed by next- statement.
- ✓ if expression is false(0), if block is by-passed and execute next- statement.
- ✓ if block with single statement the opening brace({) and closing brace(}) are optional.
- ✓ if block with compound statement(more than one statement), these set of statements must be enclosed in braces.

Control Statements

Flowchart for simple if



Control Statements

if - else statement

- ✓ This is used for two way decision making statement.
- ✓ It executes ***if*** block, when the conditions is true.
- ✓ It executes ***else*** block, when the condition is false.
- ✓ This statement mainly used to test the condition and pick one of the block.
- ✓ the ***if block*** and ***else block*** are mutually exclusive.
- ✓ if and else block with compound statement(more than one statement), these set of statements must be enclosed within the braces.
- ✓ The general syntax of ***if else*** block are

Control Statements

if(expression)

st1; } if block

else

st2; } else block

next-statement;

if(expression)

{ st1; } if block

}

else

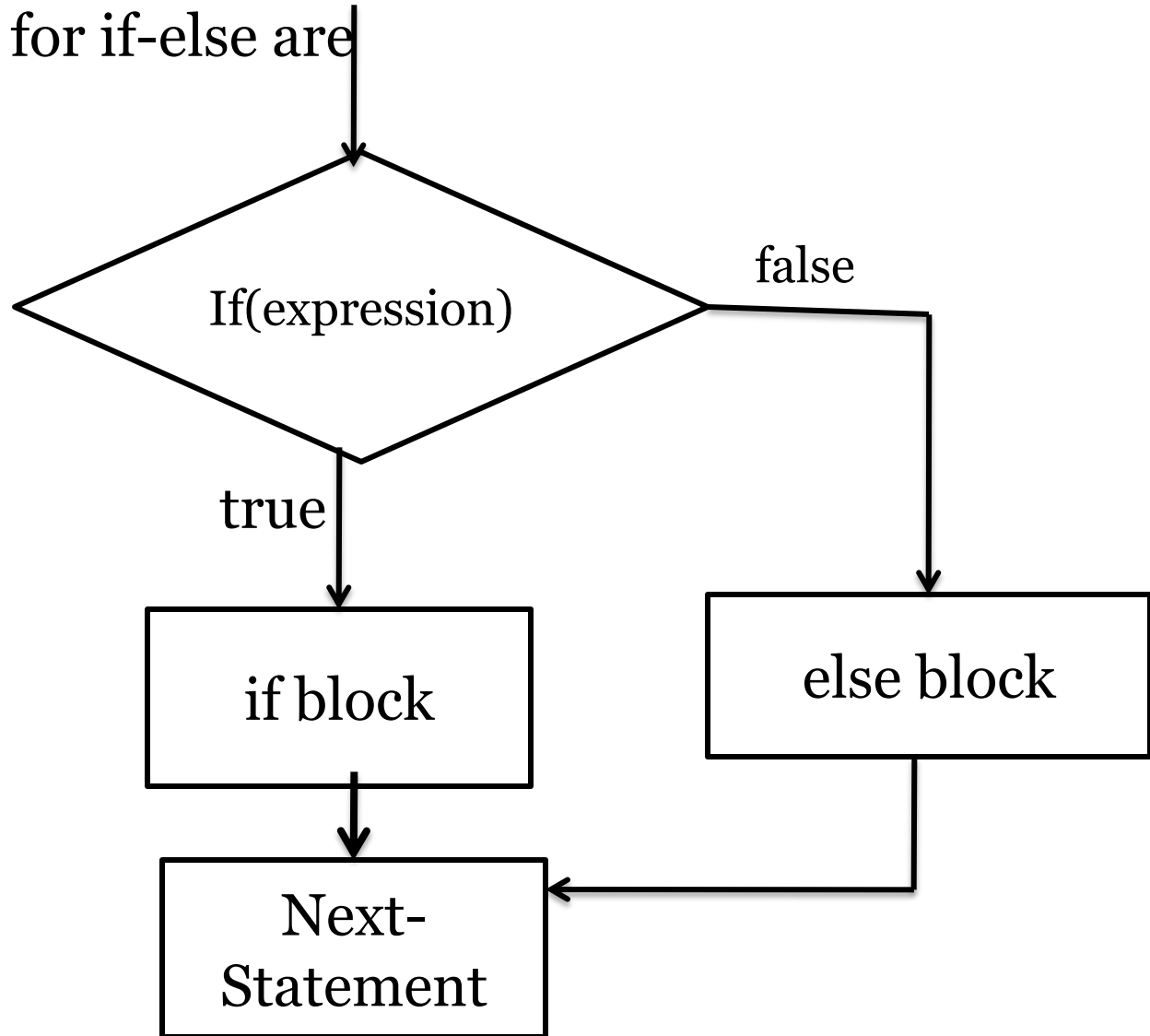
{ st1; } else block

}

next-statement;

Control Statements

The Flowchart for if-else are



Control Statements

Nested if

- ✓ To define if – else or if in another if-else statement is called nested if or nesting.
- ✓ Nesting is performed in either if block or else block or both.
- ✓ Nesting can be performed up to any level.
- ✓ The general syntax of nested if are

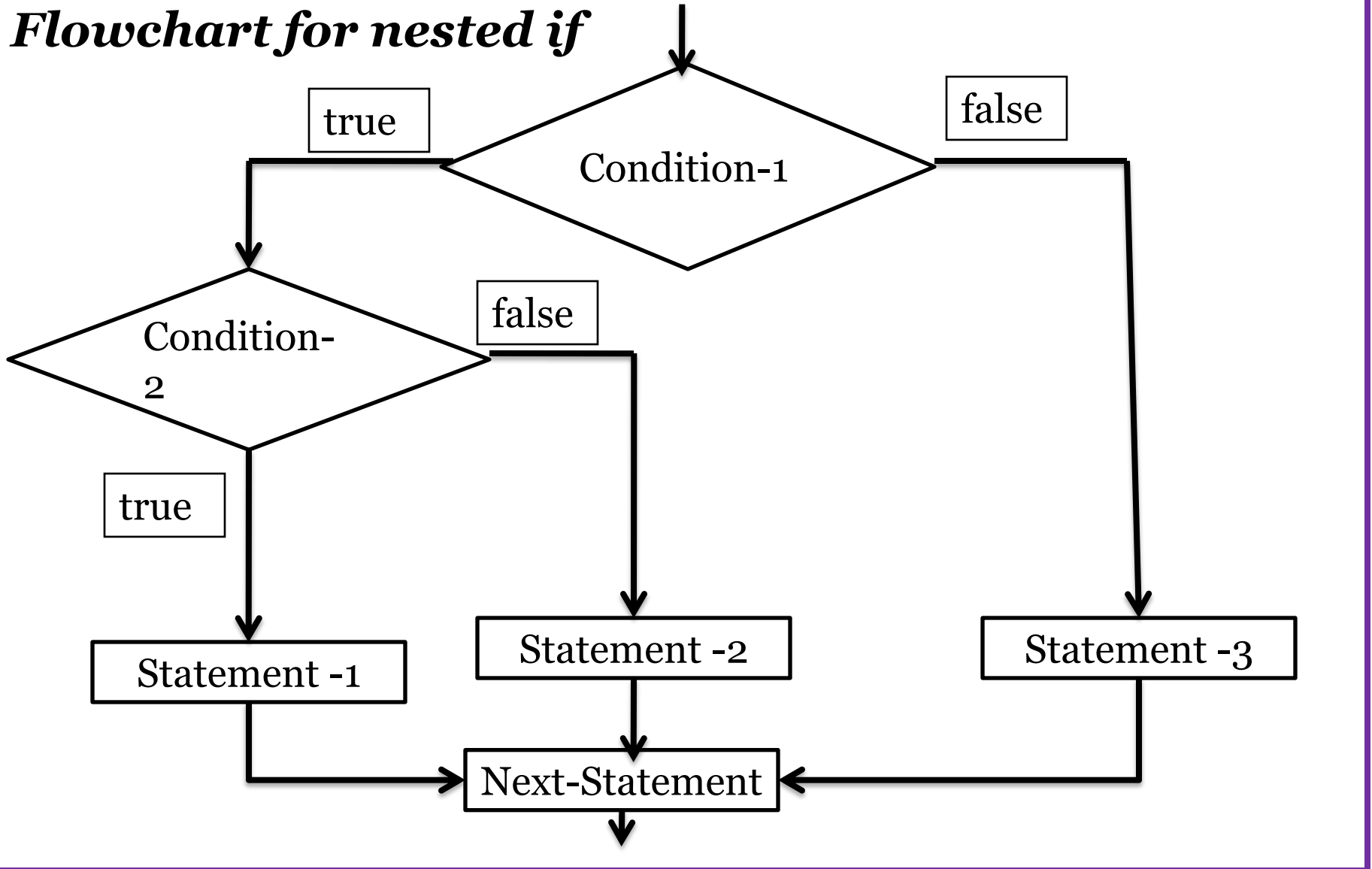
Control Statements

```
if(condition 1)
{
    if(condition 2)
    {
        st1;
    }
    else
    {
        st2;
    }
}
else
{
    st3;
}

if(condition 1)
{
    st1;
}
else
{
    if(condition 2)
    {
        st2;
    }
    else
    {
        st3;
    }
}
```


Control Statements

Flowchart for nested if



Control Statements

if – else – if ladder or else – if ladder

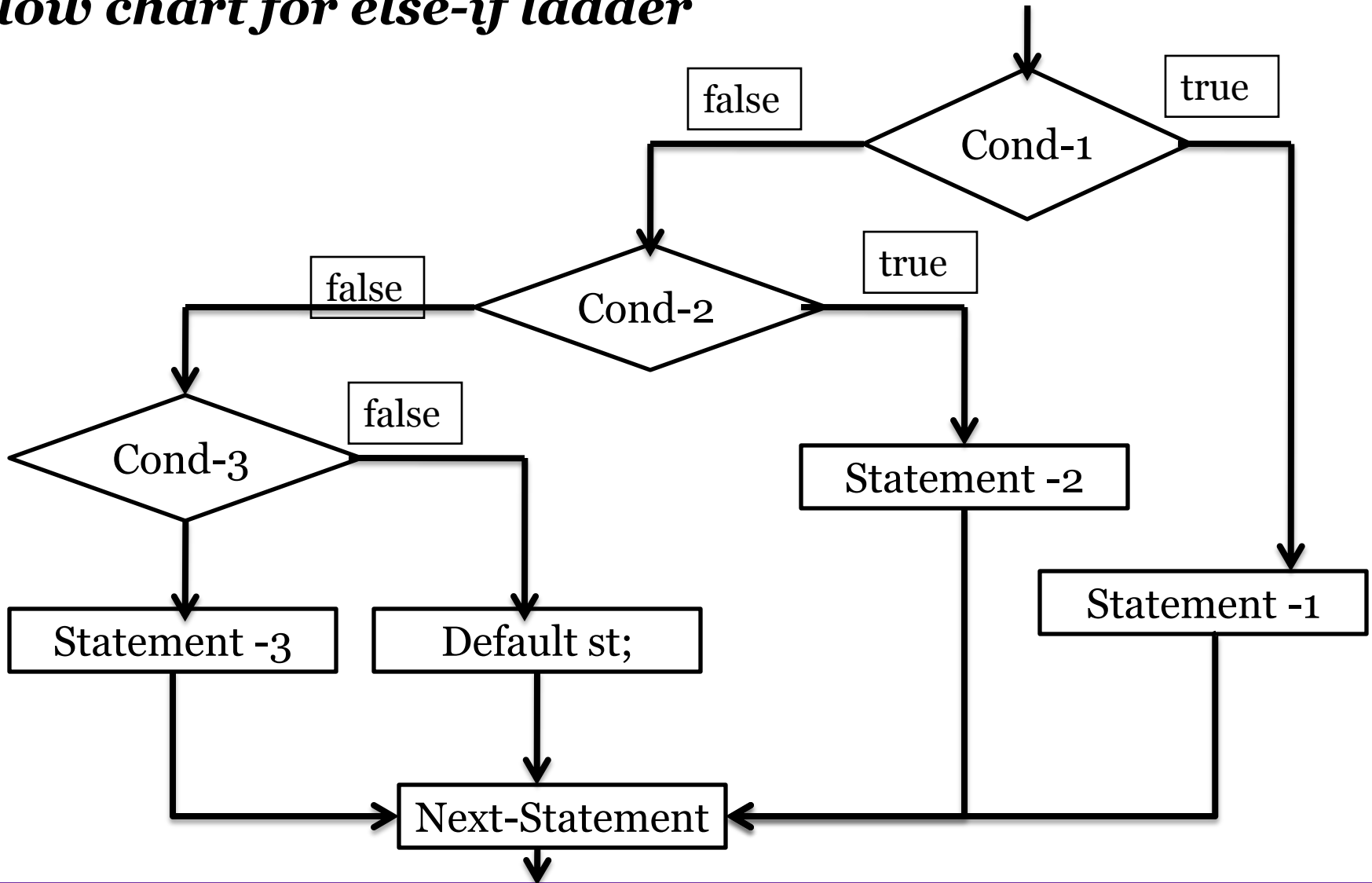
- ✓ The nested if can become quite complex, if there are more number of alternatives.
- ✓ The simplified form of nested if is ***else-if ladder***.
- ✓ This statement performs an action if condition is true, otherwise check for another condition and on which condition is true corresponding statement is executed.
- ✓ The general syntax is

Control Statements

```
if(condition -1)
    st1;
else if(cond - 2)
    st2;
else if(cond - 3)
    st3;
-----
else
    default statement;
next statement;
```

Control Statements

Flow chart for else-if ladder



Control Statements

Switch Case Statement

- ✓ The switch statement is used to execute a particular group of statements.
- ✓ It is alternative to else if ladder.
- ✓ It allows us to make a decision from the number of choices.
- ✓ It is a multi way decision statement/multi conditional statement.
- ✓ It test the expression against a list of **case** values and when a match is found, a block of statements associated with that **case** is executed.
- ✓ The general syntax of **switch-case** are

Control Statements

```
switch(expression)
{
    case int const1/char const:
        block1;
        break;
    case int const2/char const:
        block2;
        break;
    -----
    case int constN/char const:
        block N;
        break;
    default:
        default block;
}
```

Control Statements

Rules for writing switch-case statement are

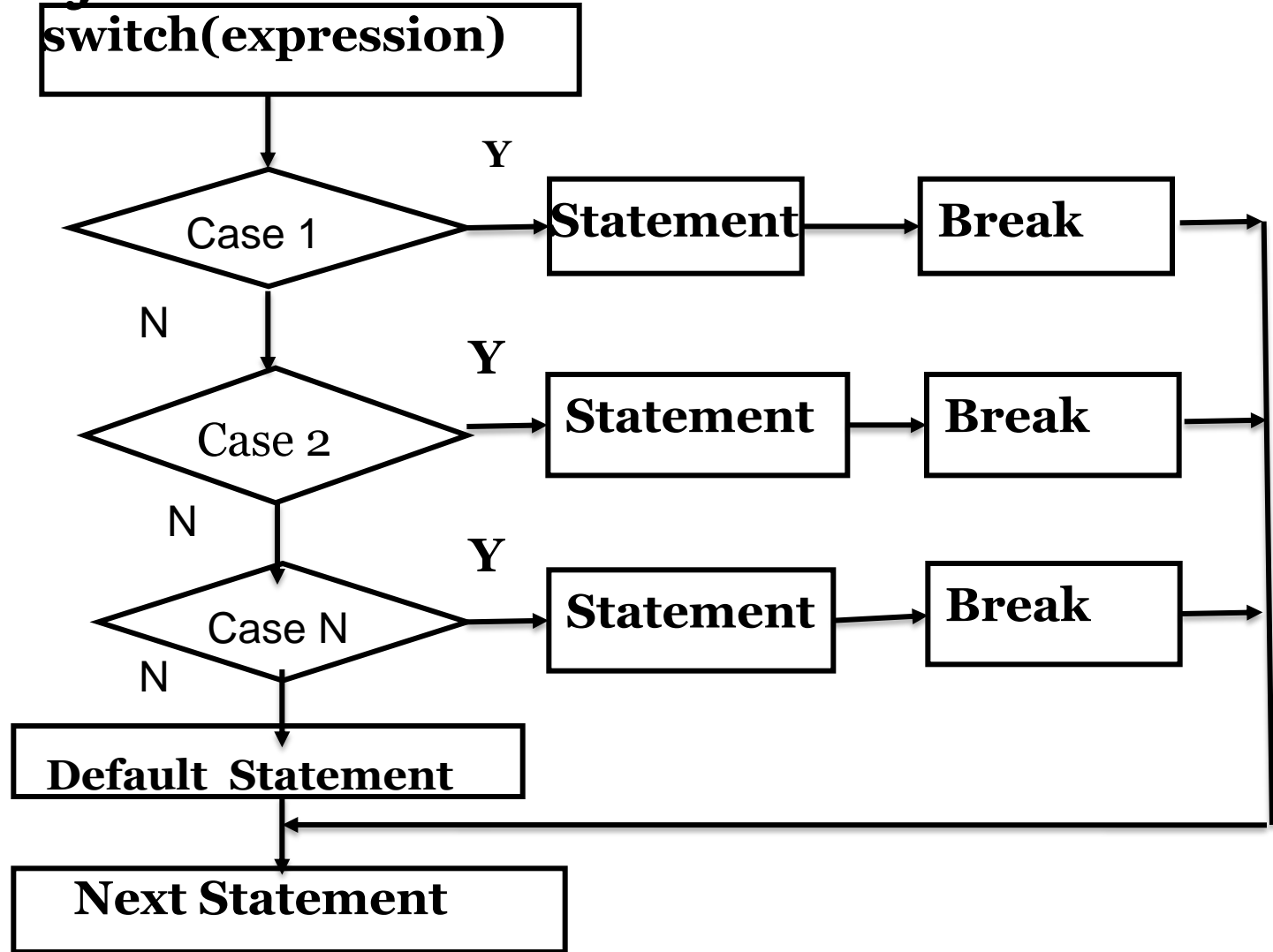
1. The result of the expression following the keyword ***switch*** must be an integer or a character.
2. No float numbers are used in expression.
3. The keyword ***case*** is followed by an integer or a character constant and operator colon(:).
4. Each case blocks are terminated with keyword ***break*** and is optional.
5. In the absence of keyword ***break***, all the cases that are followed by matched case are executed.
6. The ***break*** statement takes the control outside of the switch block.

Control Statements

7. The switch can be nested.
8. The default is optional and can be placed any where, but usually placed at end .
9. If no match is found with any of the case statements, only the default block is executed.
10. No two case constants are identical.
11. The switch statement is very useful while writing menu driven programming.

Control Statements

Flow chart for switch-case



Control Statements

Loop Control Statements / Iteration Statements

- ✓ Loop : loop is a process of executing a set statements one or more times.
- ✓ The following are the loop statements available in 'C'
 - 1) while loop
 - 2) do-while loop
 - 3) for loop
- ✓ The loop in a program consists of two parts 1. body of the loop 2.Header of the loop.

Looping process include the following steps

- ✓ Initialization of condition variable.
- ✓ Test the condition
- ✓ Execute the body of the loop depending on the condition
- ✓ Updating the condition variable.

Control Statements

While loop

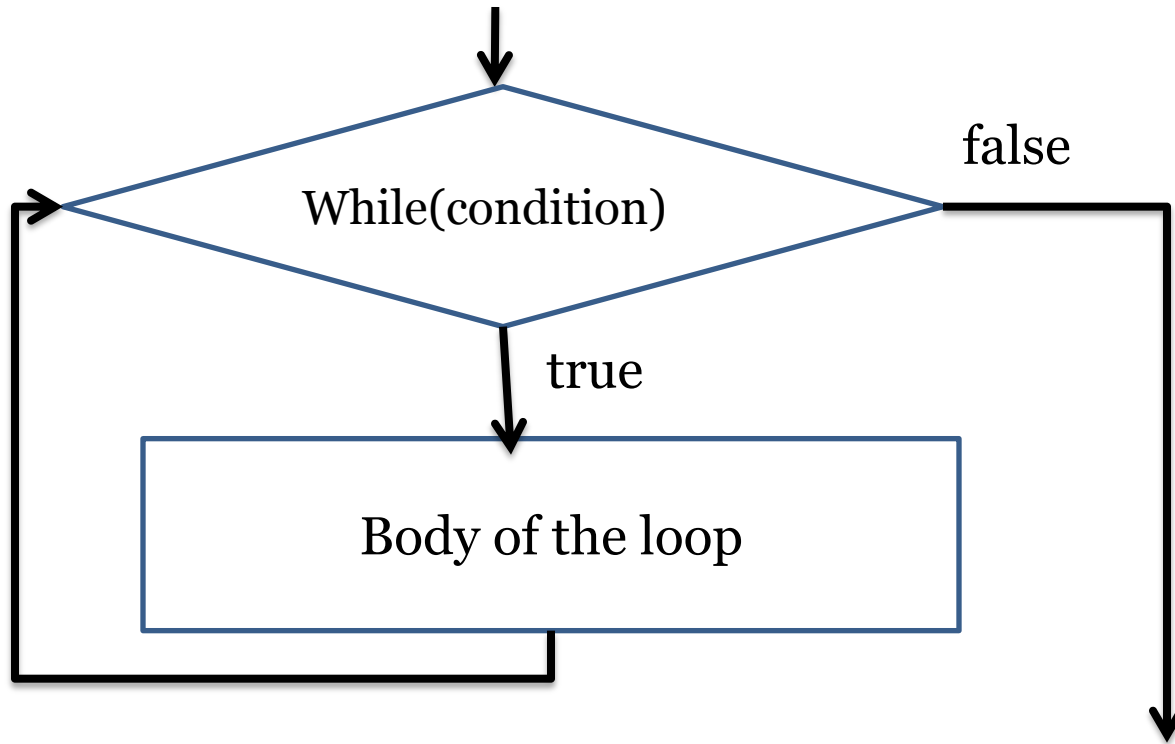
- ✓ The while loop is an entry controlled loop statement.
- ✓ It means the condition is evaluated first and if it is true, then the body of the loop is executed.
- ✓ After executing the body of the loop, the condition is once again evaluated and if it is true, the body is executed until the condition becomes false.
- ✓ If the condition false initially , the while loop executes ***zero times***.

The general syntax of while loop are

while (condition/expression)	while(expression)
	{
body	body
	}

Control Statements

Flow Chart for while-loop



Control Statements

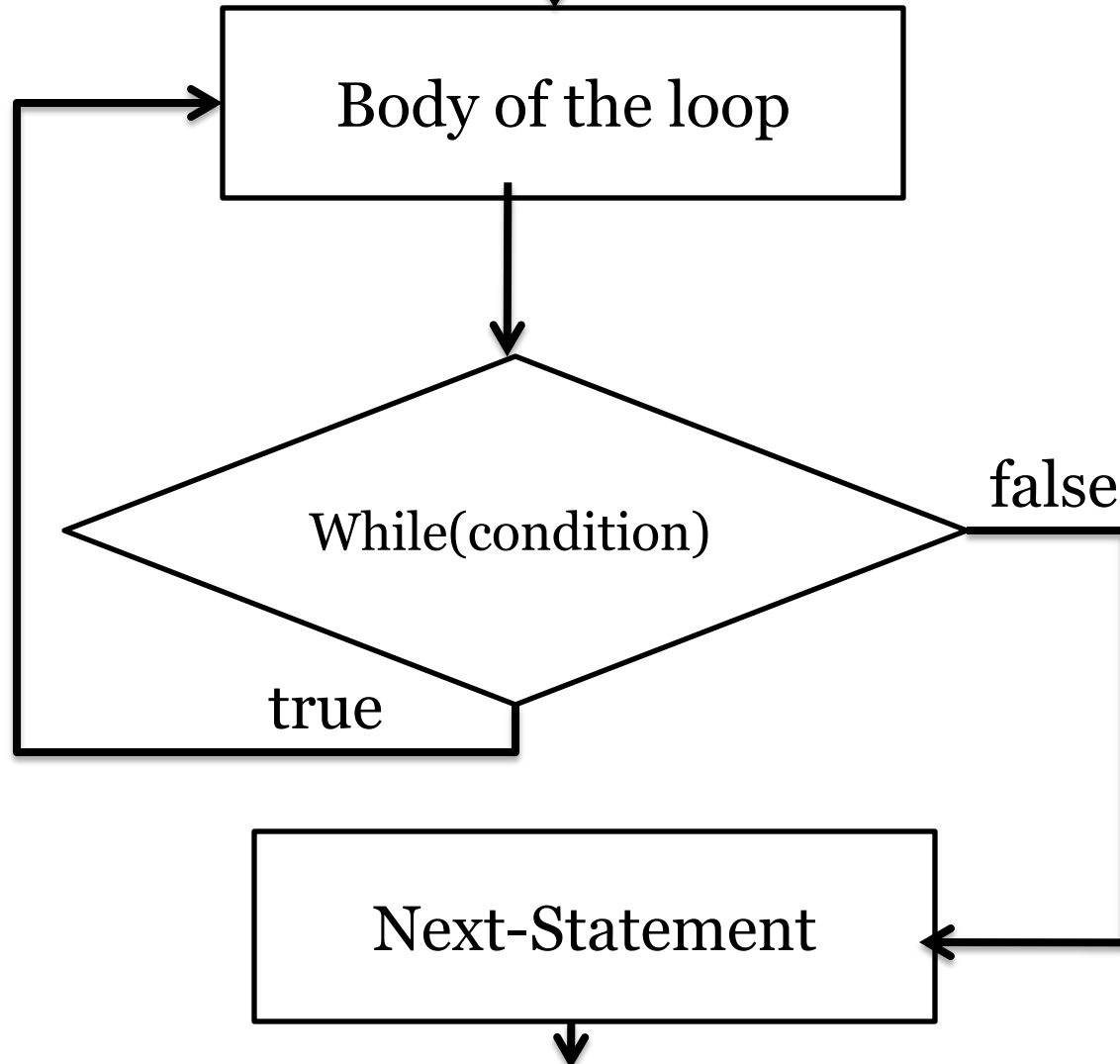
do-while loop

- ✓ do-while loop is exit controlled loop.
- ✓ do-while loop execute body of the loop and test the condition.
- ✓ The condition is true execute body of the loop once again, otherwise exit from the loop.
- ✓ If the condition false initially, do-while loop execute one time.
- ✓ The general syntax of do-while loop are

do	do
body	{
while(expression);	body
	}while(expression);

Control Statements

Flow Chart for do-while loop



Control Statements

Comparison between while and do-while loop statements

S.NO	While	Do-while
1	This is the top tested loop	This is bottom tested loop
2	The condition is first tested, if the condition is true then the block is executed until the condition becomes false.	It executes the body once, after it check the condition, if it is true the body is executed until the condition becomes false
3.	Loop will not be executed if the condition is false.	Loop is executed at least once even though the condition is false.

Control Statements

for loop statement

- ✓ the **for** is simple & most popularly used loop statement.
- ✓ The for loop allows to specify three things about a loop in a single line.
- ✓ the three things are
 - 1) initialize loop counter:** to initialize loop counter variable.
 - 2) Testing the condition:** for executing the body.
 - 3) increment/decrement counter:** is used to inc/dec loop counter.

Control Statements

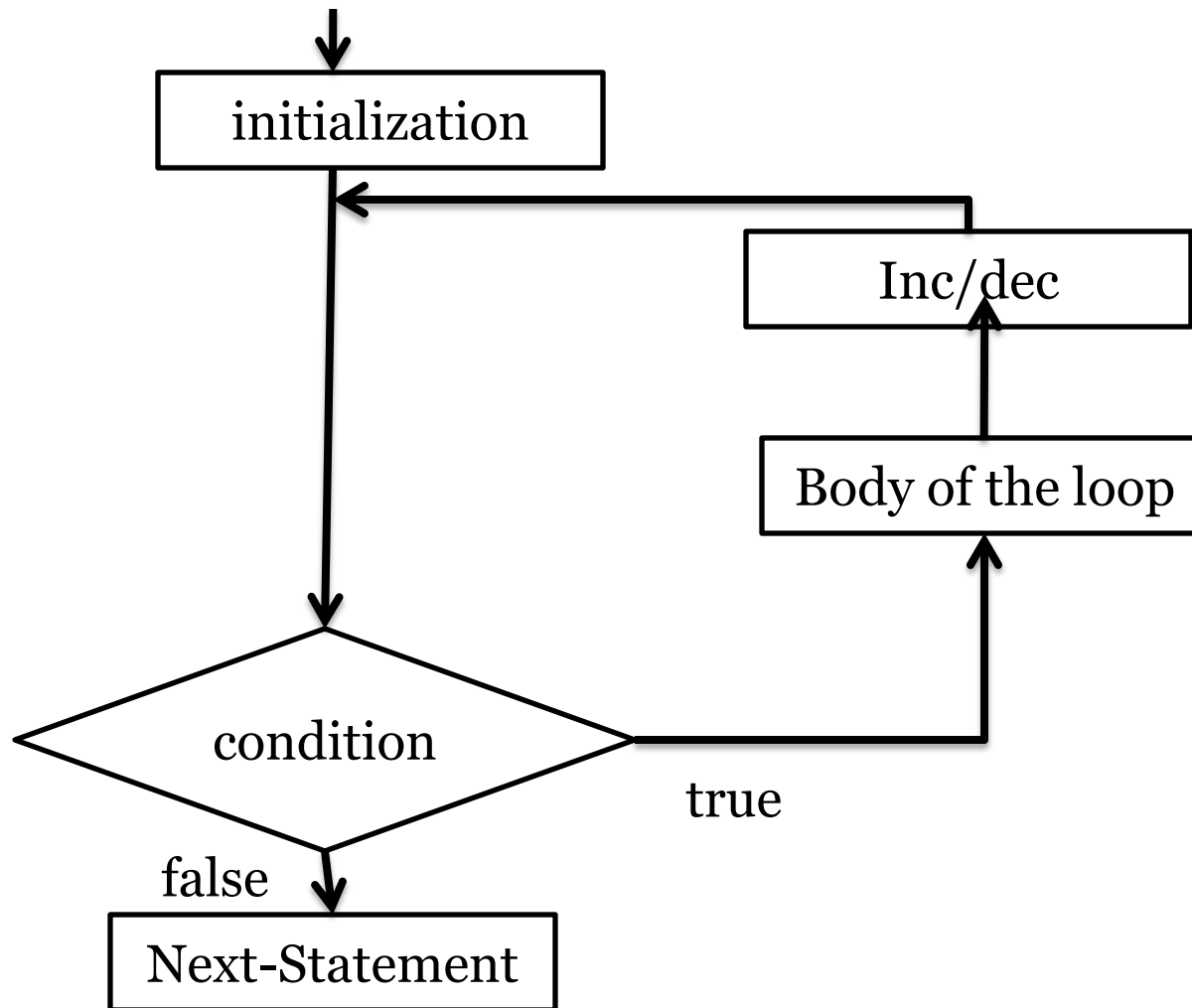
General syntax of for loop statement are

```
for(initialize counter; test condition; inc/dec)
{
    body of the loop
}
```

- ✓ Initialize counter execute exactly once at the beginning of the loop.
- ✓ After initialization, condition check is performed, if it is true body of the loop is executed.
- ✓ After executing the body of the loop, to perform inc/dec of the loop counter variable.
- ✓ Condition check, body of the loop and inc/dec are performed repeatedly until the condition becomes false.

Control Statements

Flow chart of the for loop are



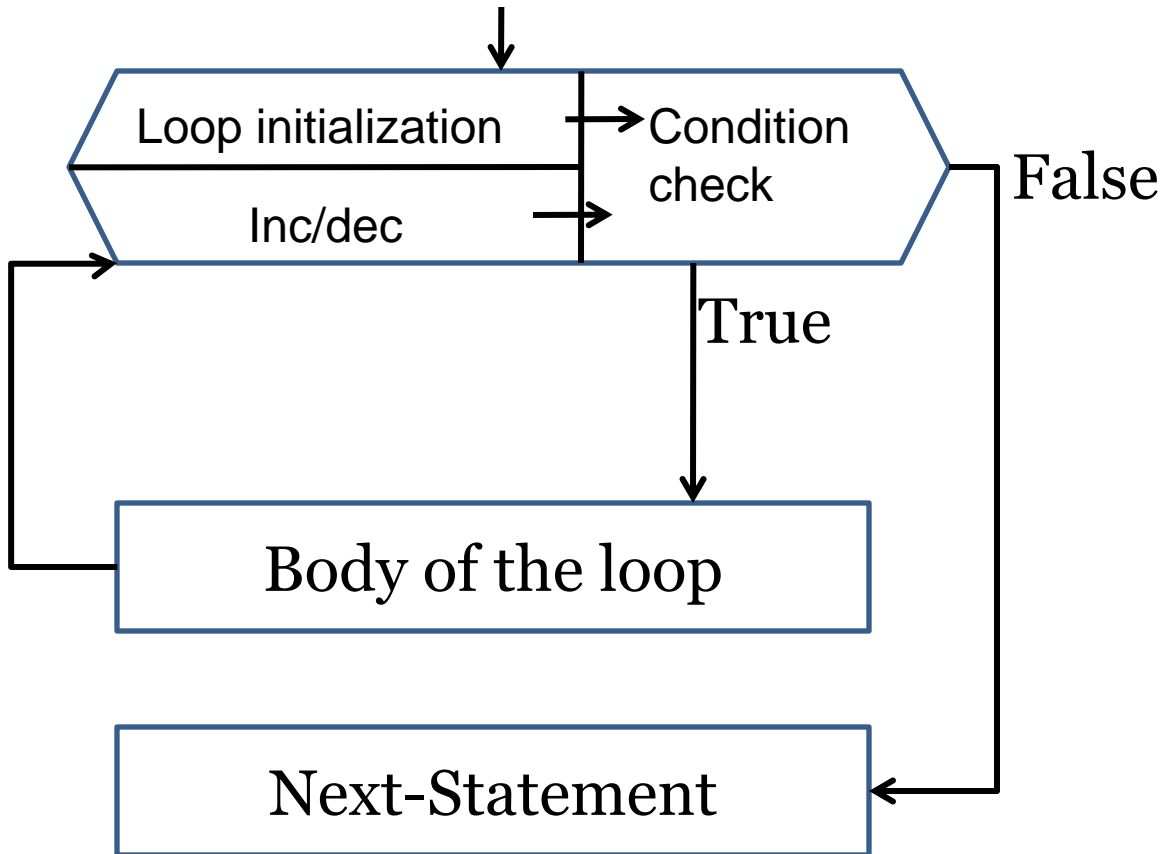
Control Statements

Additional Features of *for* loop

- ✓ we can initialize more than one variable at a time.
- ✓ For Ex: `for(i = 1,j = 1; i <= 10; i++)`
- ✓ We can inc/dec more than one variable in the inc/dec section
- ✓ For Ex: `for(i = 1,j = 1; i <= 10; i++,j++)`
- ✓ Test the condition may be any compound relation
- ✓ For Ex: `for(i = 1,j = 1; i <= 10&& j <=10; i++)`
- ✓ It may be expression in the initialization section and inc/dec section.
- ✓ For Ex: `for(a = i+j; a <= 10; a=a/2)`
- ✓ One or more sections of the for loop are omitted, if necessary.
- ✓ For Ex: `for(; i <= 10;)`

Control Statements

for loop symbol used in flowchart are



Control Statements

Jump Statements/unconditional statements

- ✓ The jump statements are used to transfer control from one place to another place randomly.
- ✓ There are four different jump statements
- ✓ break statement.
- ✓ continue statement.
- ✓ goto statement.
- ✓ return statement
- ✓ First two are legal jump statements and third are illegal jump statement.
- ✓ Legal means jumping performed logically.
- ✓ Illegal means jumping performed illogically

Note: To develop program using ***goto*** is poor programming.

Control Statements

break statement

- ✓ The ***break*** statement is used to terminate the loop during the execution.
- ✓ The ***break*** is a keyword.
- ✓ When the keyword ***break*** is used inside any loop/switch, control automatically transferred to the first statement after the loop.
- ✓ A ***break*** is usually associated with an ***if*** statement.
- ✓ General syntax is

break;

Control Statements

✓ For Ex:

```
while(condition)
```

```
{
```

```
-----
```

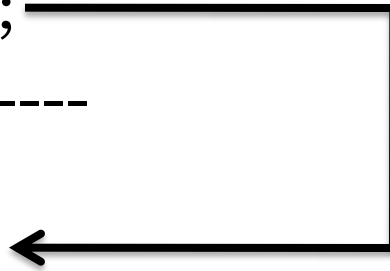
```
if(condition)
```

```
break;
```

```
-----
```

```
}
```

```
next-St;
```



Control Statements

continue Statement

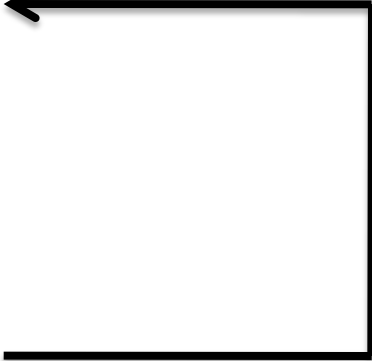
- ✓ The ***continue*** statement are used to by-pass set of statements during the execution based on the condition.
- ✓ The continue is a ***keyword***.
- ✓ A ***continue*** is usually associated with an ***if*** statement.
- ✓ When the ***continue*** is encountered inside any loop, control automatically passes to the beginning of the loop and perform next iteration.
- ✓ The general syntax is
 continue;

Control Statements

✓ For Ex: `while(condition)` ←
 {

 `if(condition)`
 `continue;`

 }



Control Statements

Difference between break & continue

S.N	break	continue
1	break statement takes the control to the out side of the loop	continue statement takes control to the beginning of loop
2	It can also be used in switch statement	This can be used only in loop statements
3	It associated with if and switch-case statements	It associated with if statement only.

Control Statements

The goto statement

- ✓ goto statement transfer control unconditionally anywhere in the program.
- ✓ The goto is a keyword.
- ✓ The goto statement require a label to identify the place to move the control.
- ✓ A label is a valid identifier end with colon(:).
- ✓ The general syntax are

label : ←

goto label; _____

Control Statements

Pascal Triangle

$$nC_r = nC_{r-1} (n - r + 1) / r$$

Algorithm for Pascal Triangle

1. Start
2. Read rows(no. of rows)
3. n-> 0
4. While(n< rows)
 - 4.1: r <- 0
 - 4.2:while(r<=n)
 - if(r=0) or r== n)
 - result <- 1

Control Statements

else

result \leftarrow resultX(n-r+1)/r

4.3: print result

5. stop.