Object Oriented Programming using C++

Introduction to C++

- C++
  - Bjarne Stroustrup (Bell Labs, 1979)
  - started as extension to C
  - added new useful, features
  - nowadays a language of its own
  - Object oriented language

General form of a C++ program

```
// Program description
#include directives
int main()
{
      constant declarations
      variable declarations
      executable statements
      return 0;
}
```

C++ keywords

- Each keyword has a predefined purpose in the language.
- Do not use keywords as variable and constant names!!

- Examples of keywords in are following:

```
bool, break, case, char, const, continue, do,
default, double, else, extern, false, float,
for, if, int, long, namespace, return, short,
static, struct, switch, typedef, true,
unsigned, void, while
```

Identifiers in C++

- Any user defined name given to:
  - Variables
  - Objects
  - Methods
  - Arrays , etc
- Rules:
  - No keywords can be used
  - Made up of alphabets, digits and underscore
  - Must start with _ or alphabet
  - Identifiers are case sensitive: d and D are different

Variables in C++

- Place to hold/store some value
- A data name given to memory location where value is stored
- Values can change during execution of program
- Example:
  - x = 10
  - Data = 45

Comments in C++

- Single line comments // from here to end of line
  - when // encountered, remainder of line ignored
  - works only on that line
  - Examples:

```
int i;  // this is a loop variable
double salary; //variable declaration
```

- Multi-line comments /* whatever in between */
  - Can cover multiple lines between the pair /* and */
  - Example:

```
/* I am a
   multiline comment */
```

Input and Output in C++

- cout (like printf in C)
  - To print some text or value on output screen
  - Used with << (insertion operator)

- cin (like scanf in C)
  - To take some input from keyboard
  - Used with >> (extraction operator)

Sample C++ Program

```
#include <iostream>
using namespace std;

int main()
{
   cout << "Hello Friend" << endl;
   cout << "How are You?";
   cout << " I am Fine";
   return 0;
}
```

Output:

```
Hello Friend
How are You? I am Fine
```

User Input

- cin >> is used
- Usage:
  - cin >> variable_name
  - Example:

```
int main() {
    int val;
    cin >> val;
    cout << val;
    return 0;
}
```

Line break

- To create a new line two options are:
  - use ‘endl’
  - use \n
- Example:

  cout << “Hello” << endl << “World”

  cout << “Hello \n World”

  Output:

  Hello

  World

Sample Program: Sum of 2 numbers

```
#include <iostream>
using namespace std;
int main()
{
    int num1, num2, sum;
   cout << "Enter first number:";
   cin >> num1;                  // num1=5
   cout << "Enter second number:";
   cin >> num2;                  //num2=3
   sum = num1 + num2;     //sum=8
   cout << "Value of addition is:" << sum;
  return 0;
}
```

Output:=>

```
Enter first number:5
Enter second number:3
Value of addition is:8
```

Thank You!

Object Oriented Programming Using C++

Topics Covered

- Data Types
- Operators
- Strings
- Conditional Constructs/Statements
  - if-else
  - switch-case
  - conditional operator

Data Types

- Every variable has some data type in C++
- Data type of a variable specifies:
  - the type of values that can be stored in the variable
  - size occupied in memory (or range of values)
  - operations that may be applied on it

- Main Primitive data types in C++ are:
  - int 4bytes (Stores Integer values) Ex: 1, -2, 67, etc.
  - float 4 bytes (Stores Floating point values) Ex: 2.1, -31.6, etc.
  - double 8 bytes (Stores Floating point values) Ex: 11.2, -2.3,etc.
  - char 1 byte (Stores a single character) Ex: ‘a’, ‘B’, ‘$’, ‘ ’,
  - bool 1 byte (Stores either of two values: true/false)

Operators in C++

- Perform some operations on variables and/or values.
- C++ has following major categories of operators:
  - Arithmetic operators +, -, *, /, %, ++, --
  - Comparison operators >, >=, <, <=, ==, !=
  - Logical operators &&, ||, !
  - Assignment operators = and its short hands +=, -=, *=, etc.
  - Bitwise operators &, |, etc.

Arithmetic Operators

- **+ addition** x=5+2 // x=7
- **- subtraction** x=5-2 // x=3
- *** multiplication** x=5*2 // x=10

- **/ division**
  - Gives quotient if both operands are integer x=5/2 // x=2
  - Gives actual division any of the operand is floating x=5.0/2.0 // x=2.5

- **% modulo division**
  - Gives remainder x=5%2 // x=1

- **++ increment** Let x=3 then, x++ makes x=4
- **-- decrement** Let x=3 then, x-- makes x=2

Sample Program: Arithmetic Operators

```
#include <iostream>
using namespace std;
int main()
{
   int x=5,y=2;
    cout << x+y << endl;       // Output is 7
    cout << x-y << endl;       // Output is 3
    cout << x*y << endl;       // Output is 10
    cout << x/y << endl;       // Output is 2
    cout << x%y << endl;       // Output is 1
    cout << 5.0/2 << endl;     // Output is 2.5
   return 0;
}
```

Output:=>

```
7
3
10
2
1
2.5
```

Comparison Operators in C++

- Compares two values and return true (1) or false (0)

•	>	**greater than**	2>5	0 (false)
•	>=	**greater than equal to**	3>=2	1 (true)
•	<	**less than**	6<8	1 (true)
•	<=	**less than equal to**	6<=6	1 (true)
•	==	**equal to**	3==4	0 (false)
•	!=	**not equal to**	3!=4	1 (true)

Sample Program: Comparison Operators

```
#include <iostream>
using namespace std;
int main()
{
   int x=5,y=2;
    cout << (x>y) << endl;        // Output is 1
    cout << (x>=y)<< endl;        // Output is 1
    cout << (x<y) << endl;        // Output is 0
    cout << (x<=y)<< endl;        // Output is 0
    cout << (x==y)<< endl;        // Output is 0
    cout << (x!=y)<< endl;        // Output is 1
   return 0;
}
```

Output:=>

```
1
1
0
0
0
1
```

Logical Operators in C++

- Combines one or more conditions
- **&& Logical AND** Returns true only if both operands are true
- **|| Logical OR** Returns false only if both operands are false
- **! Logical NOT** Invert the value

- **Examples are:**

Logical AND

3 && 4	1(true)
3 && 0	0(false)
0 && 4	0(false)
0 && 0	0(false)

Logical OR

3 || 4	1(true)
3 || 0	1(true)
0 || 4	1(true)
0 || 0	0(false)

Logical NOT

! 0	1 (true)
! 1	0 (false)
! 3	0 (false)

Sample Program: Logical Operators

```
#include <iostream>
using namespace std;
int main()
{
   int x=5,y=2;
    cout << (x&&y)<< endl;      // Output is 1
    cout << (x&&0)<< endl;      // Output is 0
    cout << (x||0)<< endl;      // Output is 1
    cout << (0||0)<< endl;      // Output is 0
    cout << (!x)  << endl;      // Output is 0
    cout << (!0)  << endl;      // Output is 1
   return 0;
}
```

Output:=>

```
1
0
1
0
0
1
```

Assignment Operator in C++

- = Assigns value to some variable e.g. x=5;
- Short Hands:
  - += x+=3 x=x+3 /
  - *= x*=3 x=x*3
  - -= x-=3 x=x-3
  - /= x/=3 x=x/3
  - %= x%=3 x=x%3

Bitwise Operator in C++

- Performs operations on corresponding bits of the operands
  - **& - Bitwise AND**
  - **| - Bitwise OR** **x EXOR y = x.y' + x'.y**
  - **~ - Bitwise NOT**
  - **^ - Bitwise EX-OR**

- **Examples:**

Bitwise AND	**Bitwise OR**	**Bitwise NOT**	**Bitwise EX-OR**
3 & 2	3 | 2	~ 2	3 ^ 2
⇒ 011	⇒ 011	⇒ ~(010)	⇒ 011
010	010	⇒ 101	010
---	---		---
010 = 2	011 = 3	= $(-3)_{10}$	001 = 1
---	---		---

Bitwise Shift Operators in C++

- Performs operations on bits of the operands
  - **<< n** - n-bits Left Shift (Doubles the value in each shift)
  - **>> n** - n-bits Right Shift (Halves the value in each shift)

- **Examples:** Let x=3

Bitwise Left Shift

x << 1 (shift 1 bit left)

⇒ 0011 ← Value of x in binary
⇒ 011 ← Shifted 1 bit left
0110 ← put 0 in LSB
= $(6)_{10}$

Bitwise Left Shift

x >> 1 (shift 1 bit right)

⇒ 0011 ← Value of x in binary
⇒ 001 ← Shifted 1 bit left
0001 ← put 0 in MSB
= $(1)_{10}$

Sample Program: Bitwise Operators

```
#include <iostream>
using namespace std;
int main()
{
   int x=3,y=2;
    cout << (x&y)  << endl;      // Output is 2
    cout << (x|y)  << endl;      // Output is 3
    cout << (~y)   << endl;      // Output is -3
    cout << (x^y)  << endl;      // Output is 1
    cout << (x<<1) << endl;      // Output is 6
    cout << (x>>1) << endl;      // Output is 1
   return 0;
}
```

Output:=>

```
2
3
-3
1
6
1
```

Strings in C++

- Sequence of characters represented between pair of double quotes
- Internally managed as objects in C++
- **E.g.:** “Hi” “Roll-1”, “#apples” “Ravi Singh”, etc.
- Need to include the header file: **<string>**

- **Creating a string variable:**

```
string name=“Ravi Singh”;
string fatherName = “Alok Singh”;
```

Strings in C++

- **Accessing string's characters:** using [index]
- First character starts from index 0.

- **Example:**

```
string name="Ravi Singh";

char firstCharacter = name[0];
char secondCharacter = name[1];
char lastCharacter = name[9];
```

Basic Operations on Strings

- **Concatenation:**
  - Joining two strings one after another
  - using ‘+’ operator

- **Example:**

```
string fName = “Ravi ”;
string lName = “Singh”;
string name = fName + lName;     //Using + operator
cout << name;                    //Output is:   Ravi Singh
```

Basic Operations on Strings

- **Append:**
  - Internally managed as objects and hence contains some functions that may be applied on string objects
  - For appending, the **append()** function can be used

- **Example:**

```
string fName = “Ravi ”;
string lName = “Singh”
string name = fName.append(lName);
cout << name;            //Output is:   Ravi Singh
```

Basic Operations on Strings

- **Finding length of a string:**
  - Length of a string is the total number of characters in the strings
  - Functions: **length()** and **size()** can be used

- **Example:**

```
string name = “Ravi Singh”;
string myText = “I am a C++ Programmer”
cout << name.length();      //Output is:  10
cout << myText.size();      //Output is:  21
```

Sample Program on Strings

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
   string fN = "Ravi ";
   string lN = "Singh";
   string name, newName;
   name = fN + lN;
   cout << name << endl;                    // Ravi Singh
   cout << name.length() << endl;           // 10
   newName = name.append(" Duggal");
   cout << newName << endl;                 // Ravi Singh Duggal
   cout << newName.size()                   // 17
   return 0;
}
```

Input Strings from User

- **cin:** Reads only first word of entire string
- **Example:**

```
string name;
cout << "Enter your name:   "
cin >> name;        // Suppose user inputs:  Ravi Singh
cout << name        // Output is: Ravi
```

- **getline():** Reads entire string containing spaces
- **Example:**

```
string name;
cout << "Enter your name:   " // Let user inputs:  Ravi Singh
getline(cin, name)  "
cout << name        // Output is: Ravi Singh
```

Conditional Statements

- Test conditions and execute a set of instructions from among multiple available sets of instructions, based on whether condition is true or false.
- Usually break the sequential flow of execution of program.

- C++ support following conditional constructs:
  - **Simple-if**: Used to execute a set of instructions if the test condition is true
  - **if-else:** Used to select from among two sets of instructions based on whether condition is true (if-block) or condition is false (else-block)
  - **else-if-else:** Used when a sequence of multiple conditions is to be checked one after another, if previous conditions fails.
  - **Switch-case:** Multi-way conditional construct to specify and select from among multiple blocks of instructions based on test expression.

Simple if

- **Syntax:**

```
if(condition)
{
    statement-1;
    statement-2;
      …..
     statement-N;
}
next-statements
…..
```

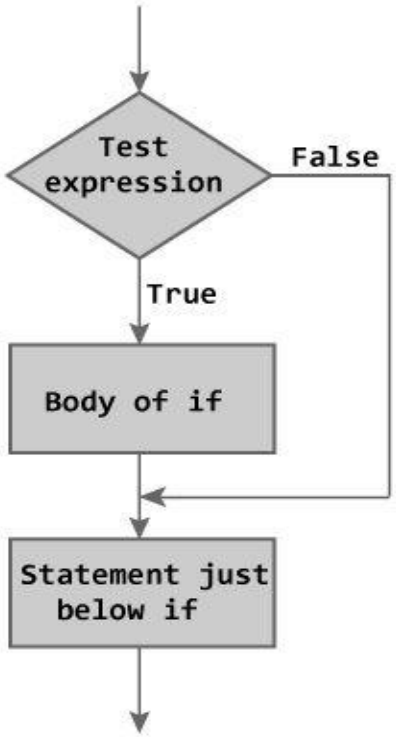


Figure: Flowchart of if Statement

- **Example:**

```
int x=3;
if(x==3)
{
   cout<<“Equal to 3”
}
cout<<“Out of if-block”
```

if-else

- **Syntax:**

```
if(condition)
{
    // True Block
    stmt-1;
      …..
    stmt-N;
}
else
{
    // False Block
    stmt-1;
      …
    stmt-M
}
next-stmts;
…..
```

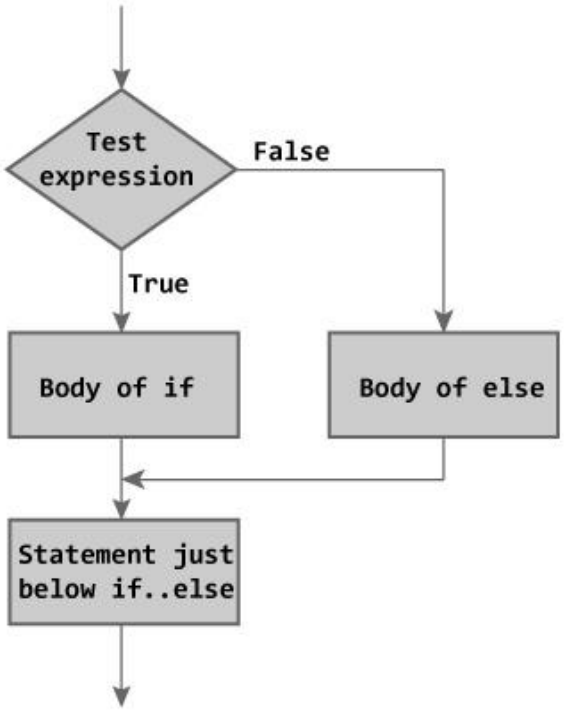


Figure: Flowchart of if...else Statement

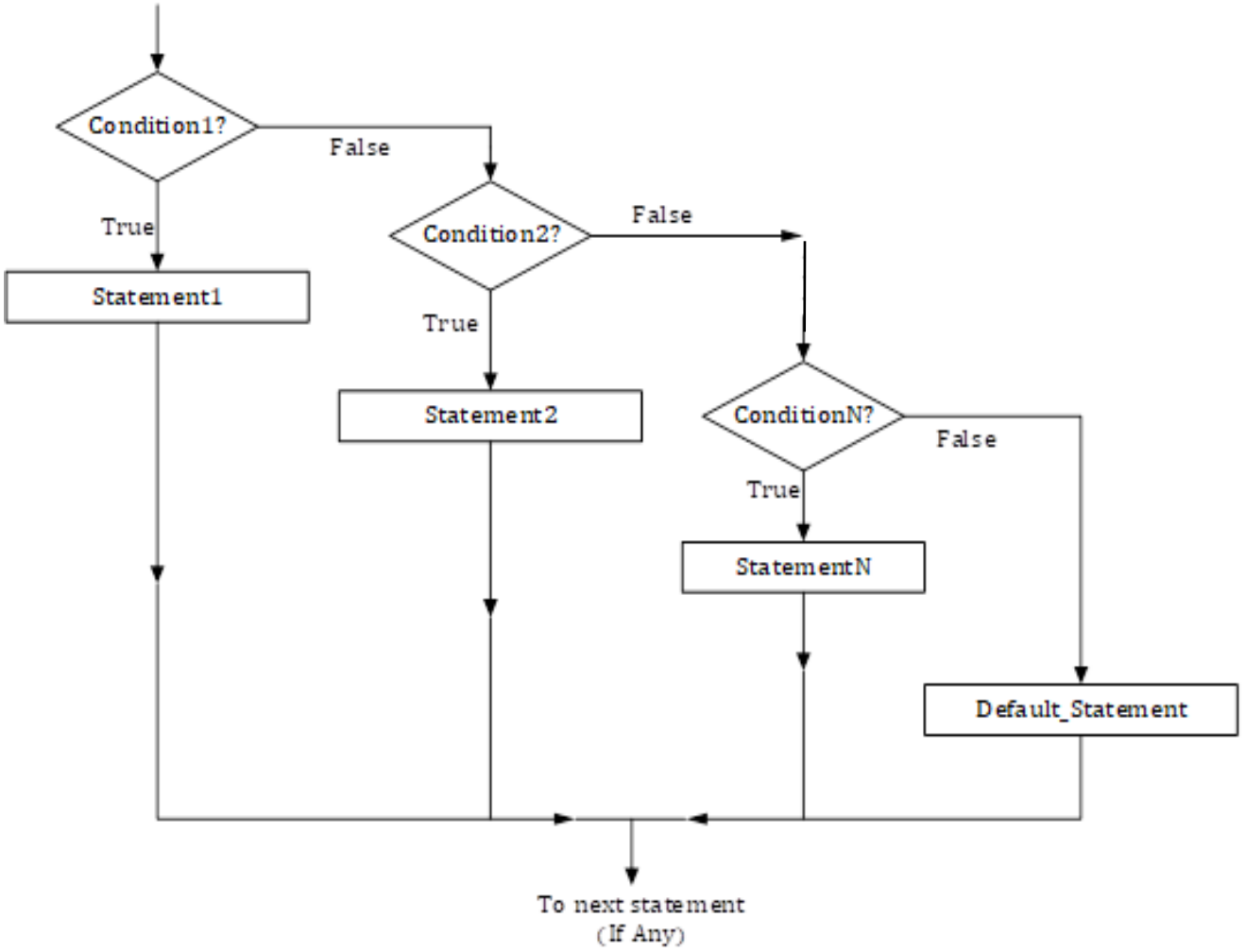
Example:

```
int age=14;
if(age>=18)
{
    cout<<“Allowed to vote”
}
else
{
    cout<<“Not Allowed”
}
cout<<“Out of if-else block”
```

else-if-else

- **Syntax:**

```
if(cond-1)
{
     /* Statements if
        cond-1 true */
}
else if(cond-2)
{
     /* Statements if
        cond-2 true */
}
else
{
    /* Statements if
       cond-1 and cond-2
       both are false */
}
next-stmts;
....
```



Sample Code

```
num1 = 15, num2=30;
if(num1>num2)
    cout<<“First number is greater”;
else if(num1==num2)
     cout<<“Both number are equal”;
else
      cout<<“Second number is greater”;
```

switch-case

- **Syntax:**

```
switch(expr)
{
      case label1: /* Statements if expr
                        matches label1 */
                   break;

      case label2: /* Statements if expr
                        matches label2*/
                   break;
      …….

      case labelN: /* Statements if expr
                      matches labelN */
                   break;

      default: /* Statements if expr does
                  not matches any label */
}
next-stmts;
....
```

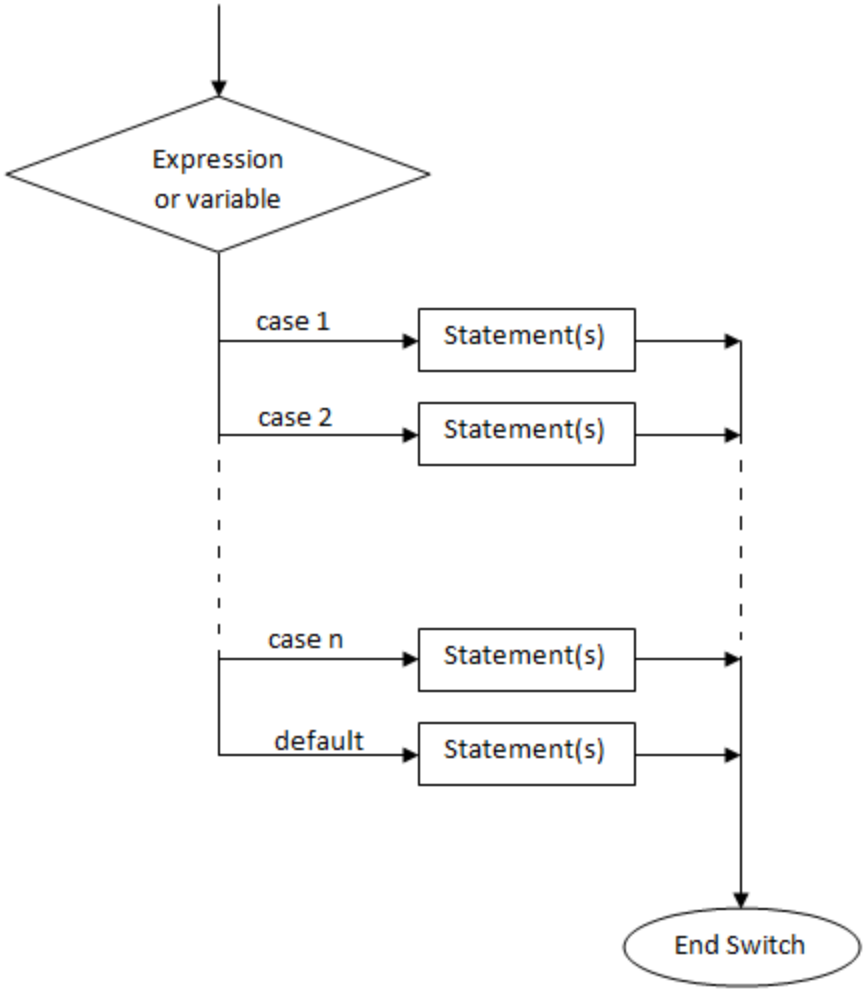


fig: Flowchart for switch case statement

Sample Code

```
int val = 2;
switch(val)
{
    case 1: cout<<“value is 1”;
                break;

    case 2: cout<<“value is 2”;
                break;

    case 3: cout<<“value is 3”;
                break;

    default: cout<<“value is not matced”;
}
cout<<“I am out of switch block”;
```

Conditional Operator [?:]

- Also known as Ternary Operator
- Work as short hand for if-else construct
- Syntax:

  condition? true-statements : false-statements

- *Example:*

  age>=18 ? cout<<“Allowed to vote” : cout<<“Not allowed”;

Thank You!