Control Statements

-Zubair uddin Shaikh

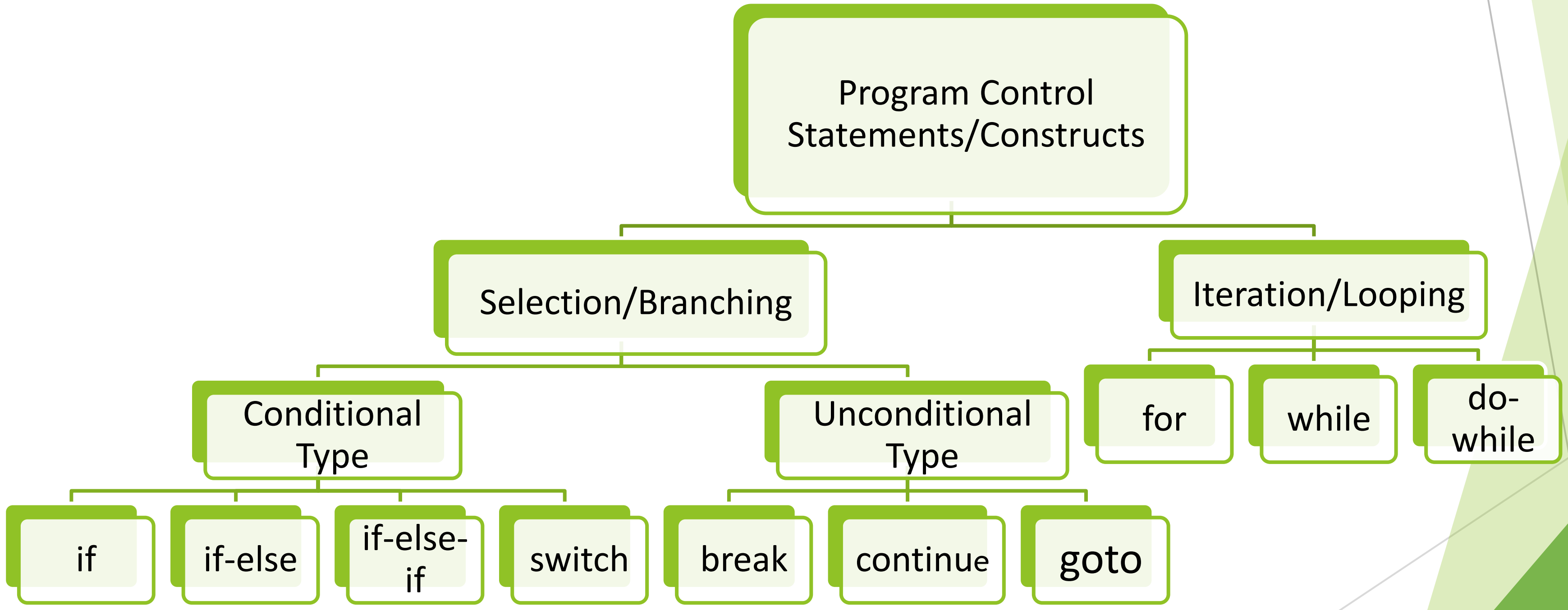
1. LEARNING OBJECTIVES

- Understanding meaning of a statement and statement block
- Learn about decision type control constructs in C and the way these are used
- Learn about looping type control constructs in C and the technique of putting them to use
- Learn the use of special control constructs such as goto, break, continue, and return
- Learn about nested loops and their utility

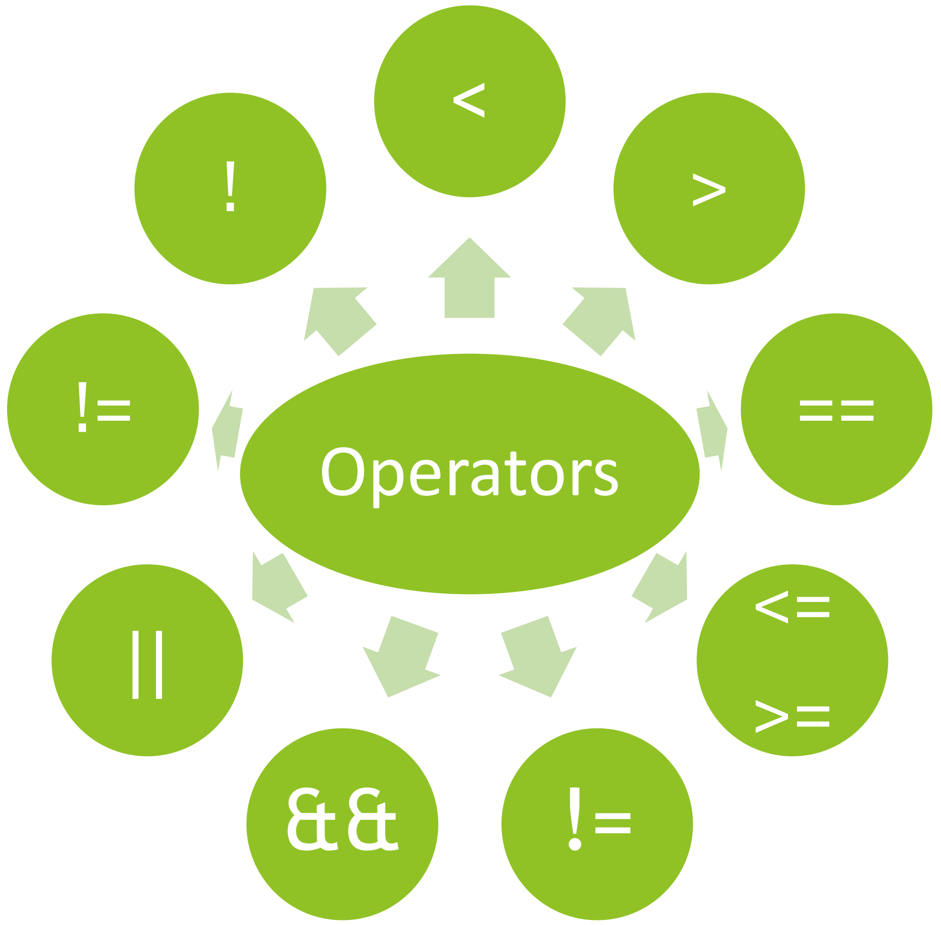
2. CONTROL STATEMENTS INCLUDE

Selection Statements	Iteration Statements	Jump Statements
• if	• for	• goto
• if-else	• while	• break
• switch	• do-while	• continue
		• return

PROGRAM CONTROL STATEMENTS/CONSTRUCTS IN ‘C’



OPERATORS



RELATIONAL OPERATORS

To Specify	Symbol Used
less than	<
greater than	>
less than or equal to greater than or equal to	<= >=

Equality and Logical Operators

To Specify	Symbol Used
Equal to	==
Not equal to	!=
Logical AND	&&
Logical OR	||
Negation	!

POINTS TO NOTE

- If an expression, involving the relational operator, is true, it is given a value of 1. If an expression is false, it is given a value of 0. Similarly, if a numeric expression is used as a test expression, any non-zero value (including negative) will be considered as true, while a zero value will be considered as false.

- Space can be given between operand and operator (relational or logical) but space is not allowed between any compound operator like <=, >=, ==, !=. It is also compiler error to reverse them.

- a == b and a = b are not similar, as == is a test for equality, a = b is an assignment operator. Therefore, the equality operator has to be used carefully.

- The relational operators have lower precedence than all arithmetic operators.

A FEW EXAMPLES

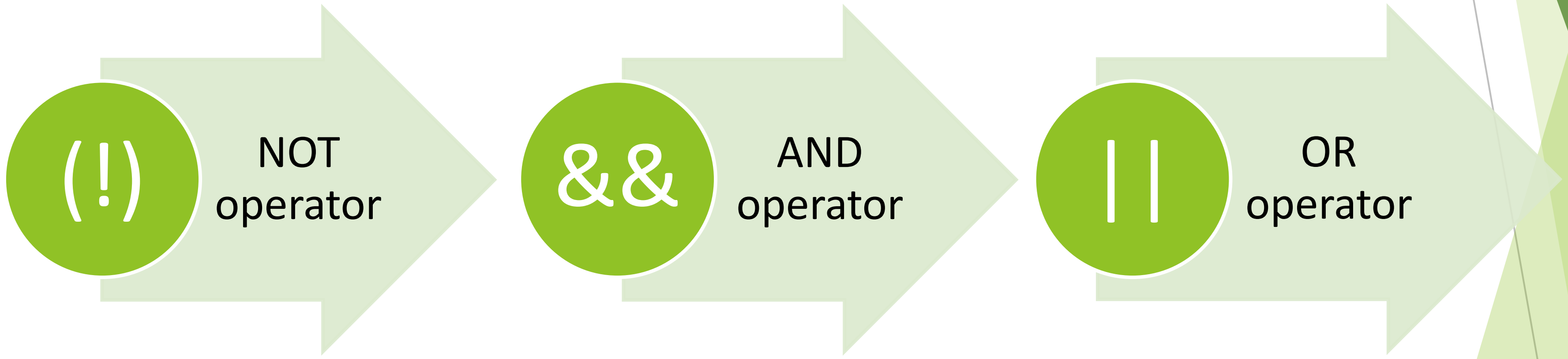
The following declarations and initializations are given:

```
int x=1, y=2, z=3;
```

Then,

- The expression x>=y evaluates to 0 (false).
- The expression x+y evaluates to 3 (true).
- The expression x=y evaluates to 2 (true).

LOGICAL OPERATORS MAY BE MIXED WITHIN RELATIONAL EXPRESSIONS BUT ONE MUST ABIDE BY THEIR PRECEDENCE RULES WHICH IS AS FOLLOWS:



OPERATOR SEMANTICS

Operators	Associativity
() ++ (postfix) -- (postfix)	left to right
+ *(unary)* - *(unary)*	right to left
++ *(prefix)* -- *(prefix)* * / %	left to right
+ -	left to right
< <= > >=	left to right
== !=	left to right
&&	left to right
||	left to right
?:	right to left
= + = - = * = / =	right to left
, (comma operator)	left to right

CONDITIONAL EXECUTION AND SELECTION

- **Selection Statements**
- **The Conditional Operator**
- **The switch Statement**

SELECTION STATEMENTS

One-way decisions using if statement

Two-way decisions using if-else statement

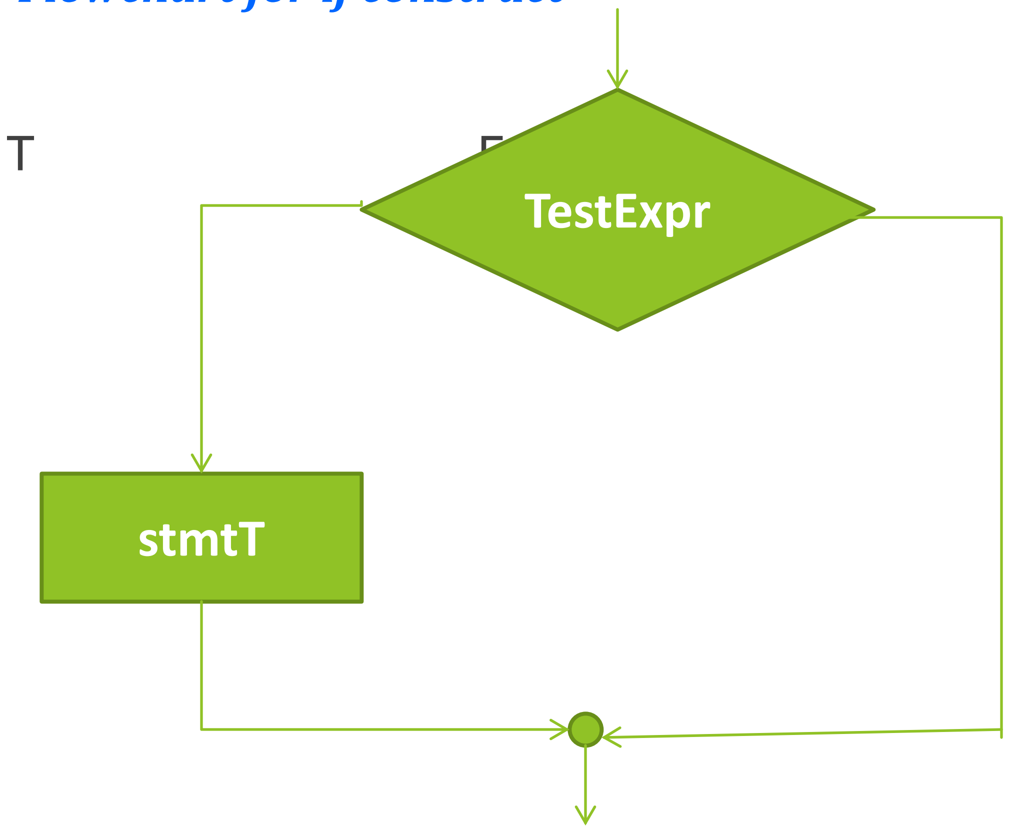
Multi-way decisions

Dangling else Problem

ONE-WAY DECISIONS USING IF STATEMENT

Flowchart for if construct

```
if(TestExpr)
stmtT;
```



WRITE A PROGRAM THAT PRINTS THE LARGEST AMONG THREE NUMBERS.

Algorithm	C Program

1. START	#include <stdio.h>
	int main()
	{
	int a, b, c, max;
	printf(“\nEnter 3 numbers”);
	scanf(“%d %d %d”, &a, &b, &c);
	max=a;
	if(b>max)
	max=b;
	if(c>max)
	max=c;
	printf(“Largest No is %d”, max);
	return 0;
	}

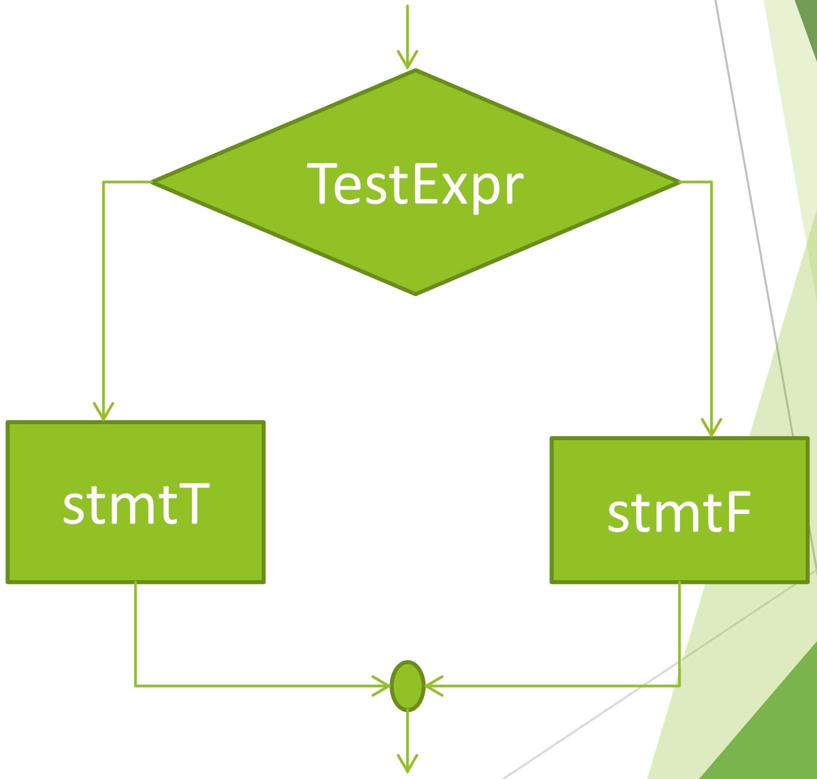
2. PRINT “ENTER THREE NUMBERS”	
3. INPUT A, B, C	
4. MAX=A	
5. IF B>MAX THEN MAX=B	
6. IF C>MAX THEN MAX=C	
7. PRINT “LARGEST NUMBER IS”, MAX	
8. STOP	

TWO-WAY DECISIONS USING IF-ELSE STATEMENT

The form of a two-way decision is as follows:

```
if(TestExpr)
  stmtT;
else
  stmtF;
```

Flowchart of if-else construct



WRITE A PROGRAM THAT PRINTS THE LARGEST AMONG THREE NUMBERS.

Algorithm	C Program

1. START	#include <stdio.h>
	int main()
	{
	int a, b, c, max;
	printf(“\nEnter 3 numbers”);
	scanf(“%d %d %d”, &a, &b, &c);
	max=a;
	if(b>max)
	max=b;
	if(c>max)
	max=c;
	printf(“Largest No is %d”, max);
	return 0;
	}

2. PRINT “ENTER THREE NUMBERS”	
3. INPUT A, B, C	
4. MAX=A	
5. IF B>MAX THEN MAX=B	
6. IF C>MAX THEN MAX=C	
7. PRINT “LARGEST NUMBER IS”, MAX	
8. STOP	

MULTI-WAY DECISIONS

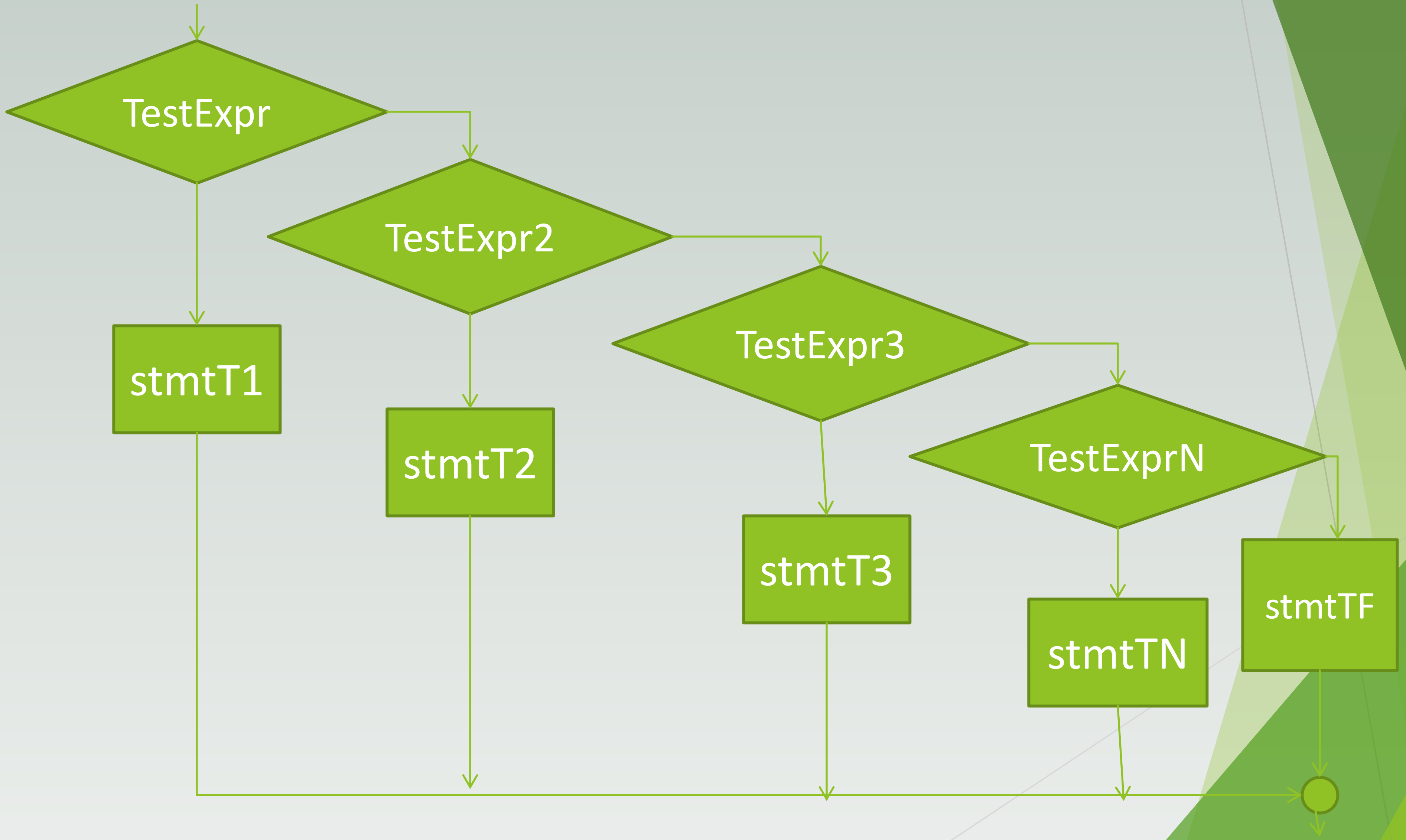
```
if(TestExpr1)
 stmtT1;
  else if(TestExpr2)
   stmtT2;
    else if(TestExpr3)
     stmtT3;
      .. .
       else if(TestExprN)
        stmtTN;
         else
          stmtF;
```

if-else-if ladder

```
switch(expr)
{
 case constant1: stmtList1;
   break;
 case constant2: stmtList2;
   break;
 case constant3: stmtList3;
   break;
    …………………………..
    …………………………..
 default: stmtListn;
}
```

General format of switch statements

FLOWCHART OF AN IF-ELSE-IF CONSTRUCT



THE FOLLOWING PROGRAM CHECKS WHETHER A NUMBER GIVEN BY THE USER IS ZERO, POSITIVE, OR NEGATIVE

```
#include <stdio.h>
int main()
{
   int x;
   printf(“\n ENTER THE NUMBER:”);
   scanf(“%d”, &x);
   if(x > 0)
       printf(“x is positive \n”);
          else if(x == 0)
            printf(“x is zero \n”);
              else
               printf(“x is negative \n”);
                return 0;
}
```

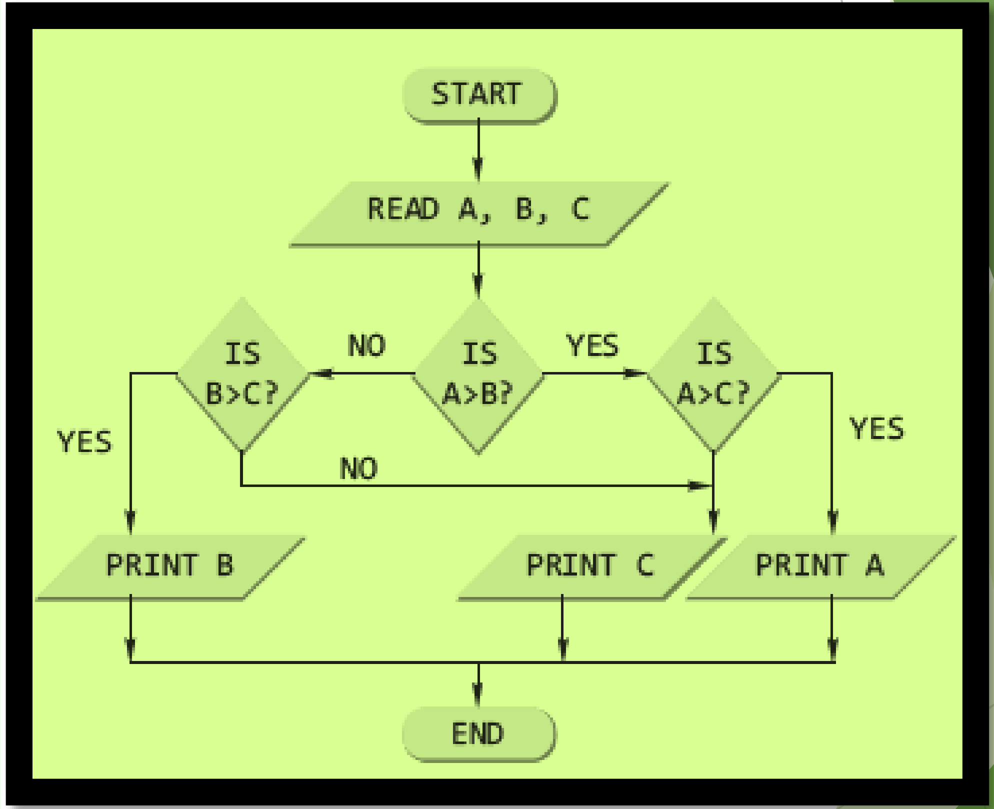
NESTED IF

- When any if statement is written under another if statement, this cluster is called a nested if.
- The syntax for the nested is given here:

Construct 1	Construct 2
`if(TestExprA)`	`if(TestExprA)`
`    if(TestExprB)`	`    if(TestExprB)`
`      stmtBT;`	`      stmtBT;`
`    else`	`    else`
`      stmtBF;`	`      stmtBF;`
` else`	`else`
`    stmtAF;`	`    if(TestExprC)`
	`      stmtCT;`
	`    else`
	`      stmtCF;`

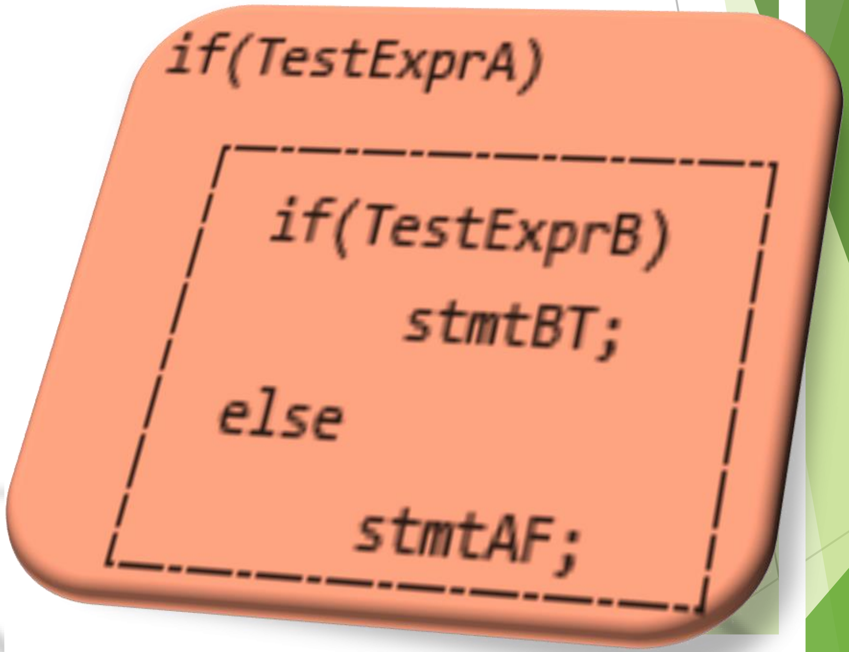
A PROGRAM TO FIND THE LARGEST AMONG THREE NUMBERS USING THE NESTED LOOP

```
#include <stdio.h>
int main()
{
  int a, b, c;
  printf("\nEnter the three numbers");
  scanf("%d %d %d", &a, &b, &c);
  if(a > b)
      if(a > c)
         printf("%d", a);
      else
          printf("%d", c);
  else
      if(b > c)
          printf("%d", b);
      else
          printf("%d", c);
  return 0;
}
```



DANGLING ELSE PROBLEM

- This classic problem occurs when there is no matching else for each if. To avoid this problem, the simple C rule is that always pair an else to the most recent unpaired if in the current block. Consider the illustration shown here.
- The else is automatically paired with the closest if. But, it may be needed to associate an else with the outer if also.



```
if(TestExprA)
    if(TestExprB)
          stmtBT;
    else
          stmtAF;
```

SOLUTIONS TO DANGLING ELSE PROBLEM

- Use of null else
- Use of braces to enclose the true action of the second if

With null else	With braces

```
if(TestExprA)

if(TestExprB)
      stmtBT;
   else
      ;
else
   stmtAF;
```
	```
	if(TestExprA)
	{

	if(TestExprB)
	    stmtBT;
	}
	else
	   stmtAF;
	```

6. THE CONDITIONAL OPERATOR

- It has the following simple format:

  expr1 ? expr2 : expr3

  It executes by first evaluating expr1, which is normally a relational expression, and then evaluates either expr2, if the first result was true, or expr3, if the first result was false.

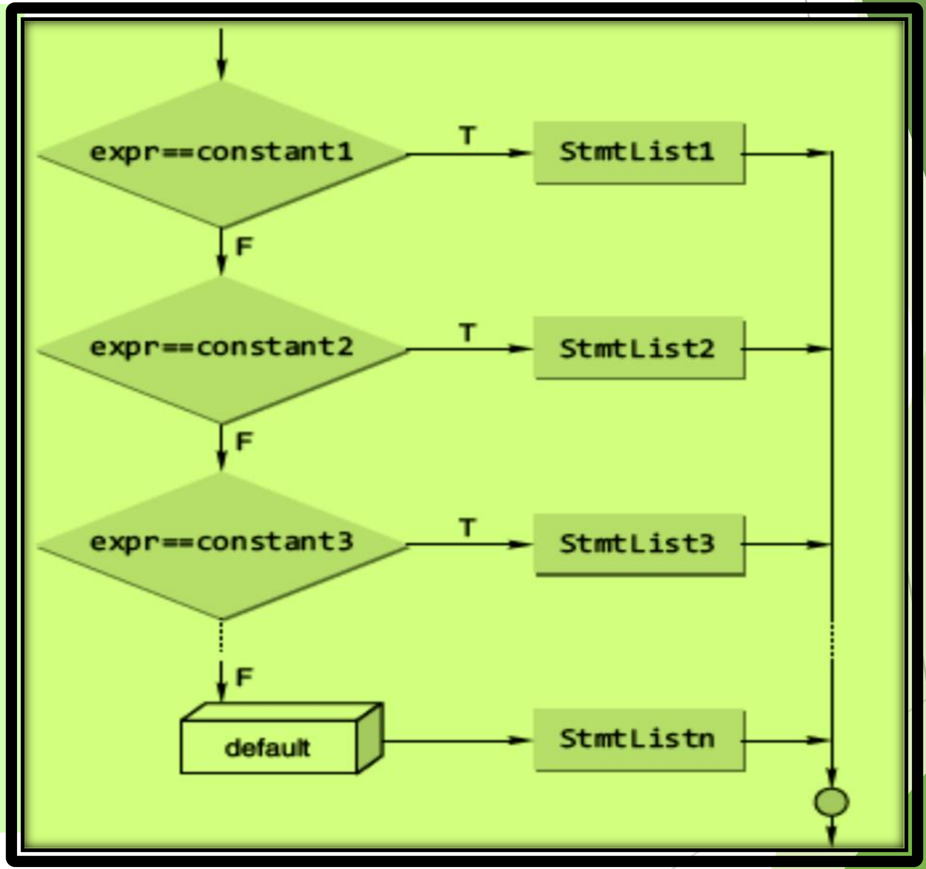
```
#include <stdio.h>
int main()
{
 int a,b,c;
 printf("\n ENTER THE TWO
     NUMBERS:");
  scanf("%d %d", &a, &b);
  c=a>b? a : b>a ? b :-1;
  if(c==-1)
       printf("\n BOTH NUMBERS ARE
     EQUAL");
  else
  printf("\n LARGER NUMBER IS %d",c);
  return 0;
}
```

An Example

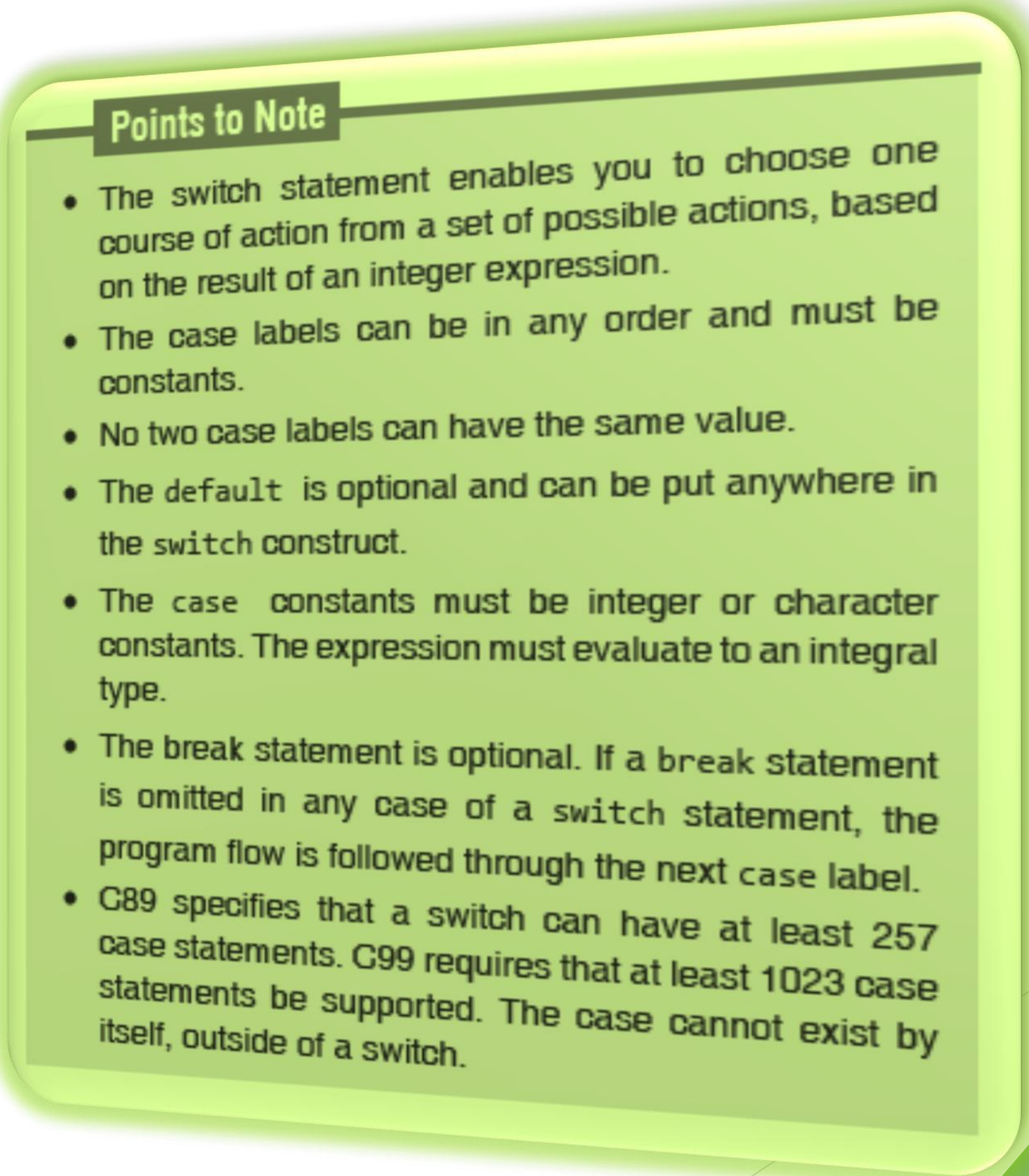
THE SWITCH STATEMENT

The general format of a switch statement is

```
switch(expr)
{
case constant1: stmtList1;
break;
case constant2: stmtList2;
break;
case constant3: stmtList3;
break;
…………………………….
…………………………….
default: stmtListn;
}
```



The C switch construct



Points to Note

- The switch statement enables you to choose one course of action from a set of possible actions, based on the result of an integer expression.
- The case labels can be in any order and must be constants.
- No two case labels can have the same value.
- The `default` is optional and can be put anywhere in the `switch` construct.
- The `case` constants must be integer or character constants. The expression must evaluate to an integral type.
- The break statement is optional. If a `break` statement is omitted in any case of a `switch` statement, the program flow is followed through the next `case` label.
- C89 specifies that a switch can have at least 257 case statements. C99 requires that at least 1023 case statements be supported. The case cannot exist by itself, outside of a switch.

SWITCH VS NESTED IF

- **The switch differs from the else-if in that switch can test only for equality, whereas the if conditional expression can be of a test expression involving any type of relational operators and/or logical operators.**

- **A switch statement is usually more efficient than nested ifs.**

- **The switch statement can always be replaced with a series of else-if statements.**