Branching statements

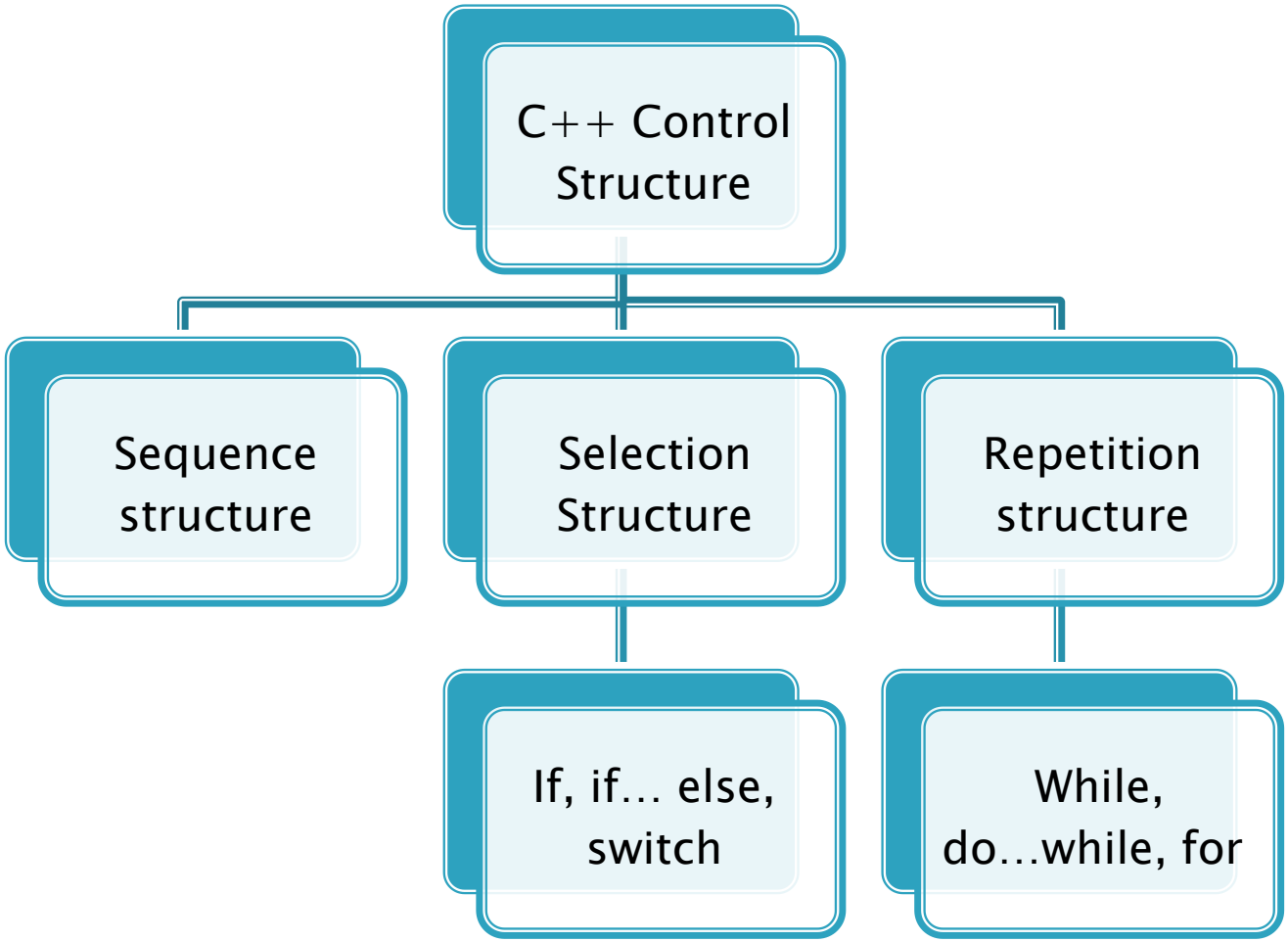
Objectives

- By the end of this section you should be able to:
  - Differentiate between sequence, selection, and repetition structure.
  - Differentiae between single, double and multiple selection statements.
  - Identify and use the C++ conditional operator (?:)
  - Use the nested control statement and explain its meaning.
  - Determine the problems that require the user of nested control structure.

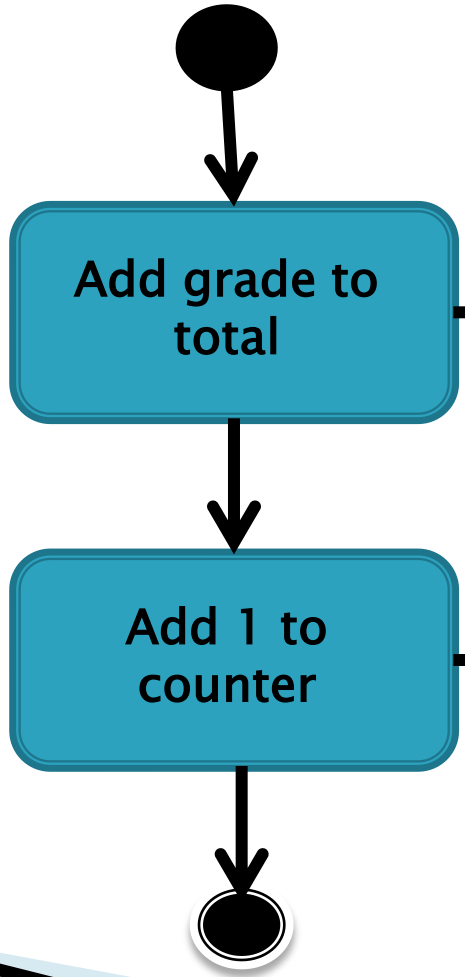
Control Structure (Logic Structure)

- Used to design data flow in modules and programs as whole
- Basic structures:
  - Sequence structure:
    - Process one instruction after another.
  - Selection structures:
    - Decision structure
      - Make choices by using relational or logical operators
    - Case structure
      - Enable to pick up one of a set of tasks
  - Repetition structure:
    - Enable to repeat tasks.

Control structures in C++



Sequence-structure Activity Diagram



Corresponding C++ statement:

```
total=total + grade;
```

Corresponding C++ statement:

```
counter=counter + 1;
```

Selection Statements

- Three types:
  - Single selection statement
    - Selects or ignores a single group of actions.
  - Double-selection statement
    - Selects between two groups of actions.
  - Multiple-selection statement
    - Selects among many group of actions.

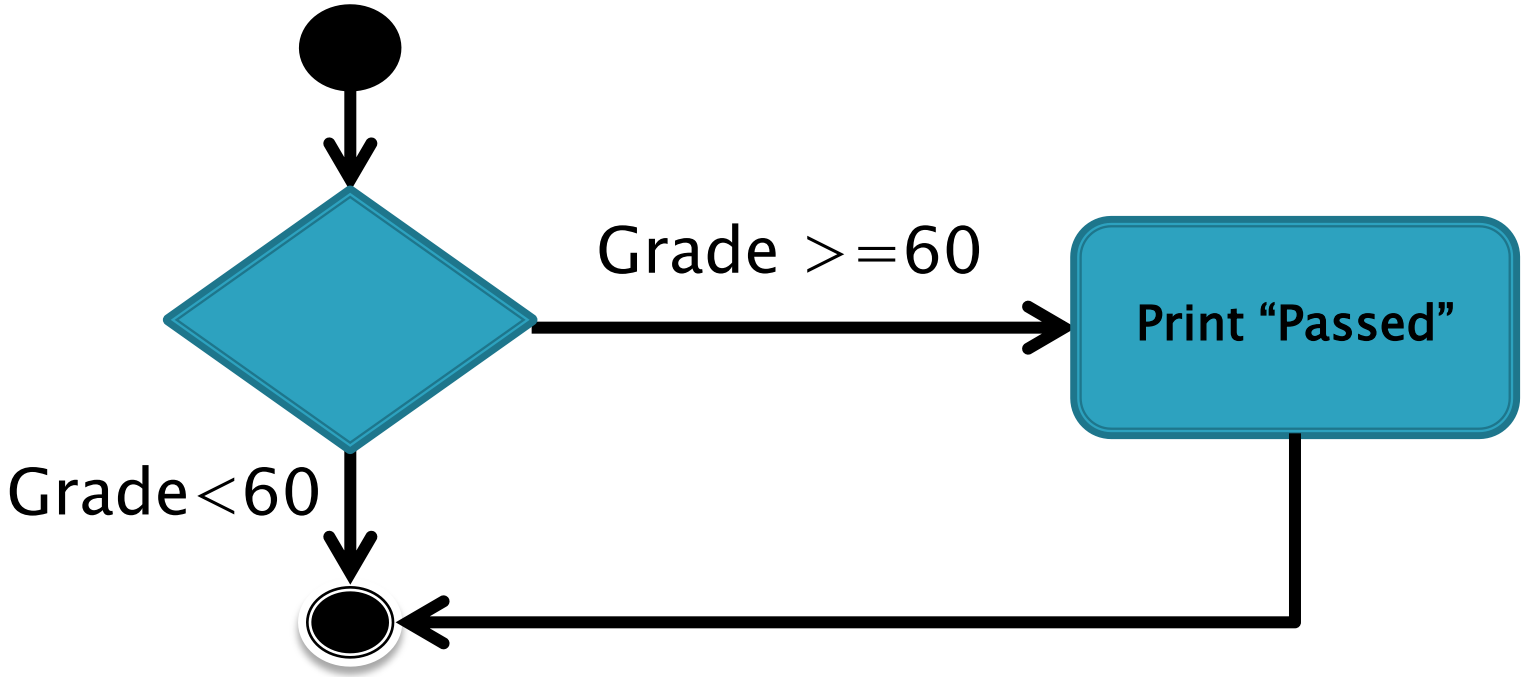
If Single Selection Statement

- Use the keyword `if`
- Begin with if followed by condition; then action or group of actions (compound statements or blocks) are listed.
- Enclose compound statements between braces {} and indent these statements
- If condition is true , action is performed. Otherwise, actions is ignored
- Example:

```
if (grade >=60)
cout<<“Passed”;
```

Pseudocode:
If student grade is greater than or equal to 60
Print “Passed”

If Single Selection Statement activity diagram



If Single selection statement

```
if (condition)
  action;
```

Condition can be relational or equality operators or any other expressions

```
if (condition)
{
  action1;
  action2;
  …..
  ….
  ….
  actionN;
}
```

if…else Double-selection Statement

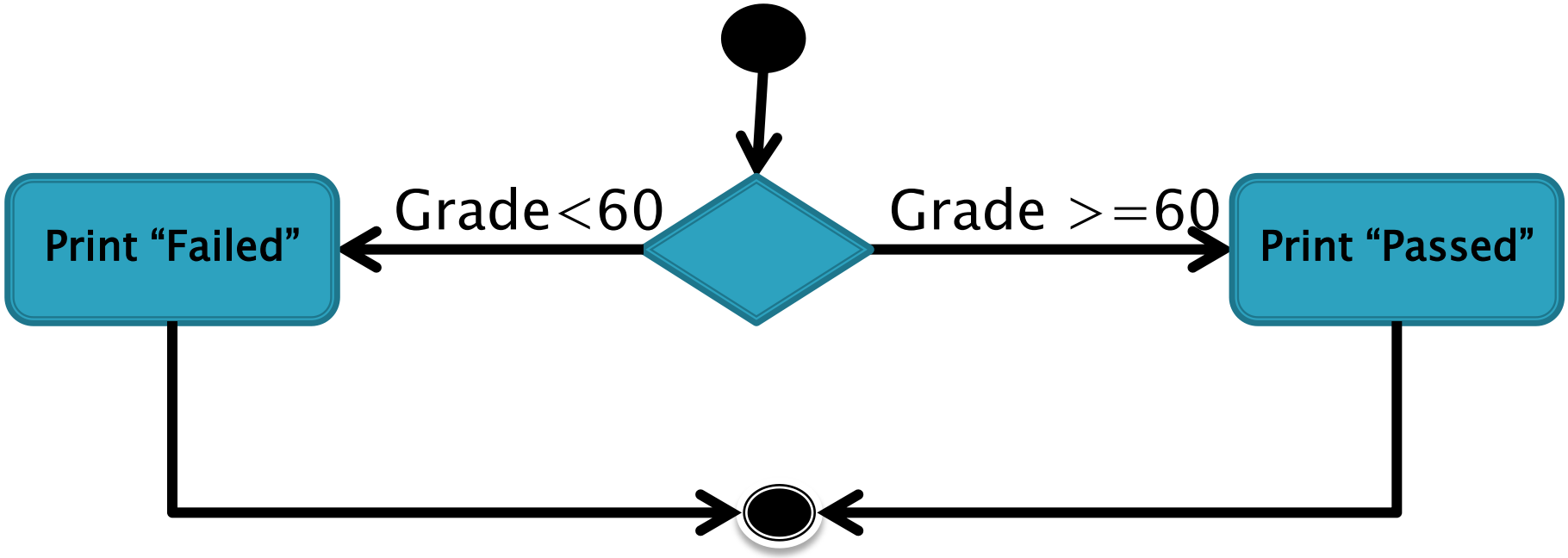
- Use the keyword `if` and `else`
- Begin with if followed by condition; then action or
  group of actions are listed.
- End with else then action or group of actions are listed.
- If condition is true, action that followed by `if` is performed. Otherwise, action that followed by `else` is performed.

Double Selection Statement

```
if (condition)
  action1;
else
  action2;
```

Condition can be relational or equality operators or any other expressions

if…else double-selection statement activity diagram



if...else Double-selection Statement

- `if...else` double-selection statement
- The following pseudocode prints "Passed" if the student's grade is greater than or equal to 60, or "Failed" if the student's grade is less than 60.

  If student's grade is greater than or equal to 60
  Print "Passed"
  Else
  Print "Failed"

- The preceding pseudocode *If...Else* statement can be written in C++ as

```
if ( grade >= 60 )
   cout << "Passed";
else
   cout << "Failed";
```

Nested if.. else Statements

- One inside another, test for multiple cases
- Once condition met, other statements skipped

Example:

```
if( n > 0 )
    if( n%2 == 1 )
        cout << " Positive odd number ";
    else
        cout << "Positive even number";
```

Conditional Operator (?:)

- Provide similar result of if…else double selection statement
- Ternary operator requires three operands:
  - The first operand is a condition
  - The second operand is the value for the entire conditional expression if the condition is `true`
  - The third operand is the value for the entire conditional expression if the condition is `false`.
- Syntax:

  Condition? Condition's true value: condition's false value

Conditional Operator (?:)

- Example
  - Grade >=60 ? Cout<<“Passed”: cout<<“Failed”;
- Can be written as:

```
cout<<(grade >=60 ? “passed”: “failed”);
```

- Example:

```
int i=1, j=2, max;
max=(i>j ?i:j);
```

Nested if.. else Statements

```
if (conditioin1)
  action1;
else
  if (condition2)
    action2;
  else
    if(condition3)
       action3;
    …

          else
                actionN;
```

Nested if … else statements

```
if (condition1)
{
   if (condition2)
     action1;

   else

     {
        if (condtion3)
        action2;
        else
        action3;
     }
}
else
   action4;
```

Nested if…else Statements

- Write the pseudocode for if…else statements that prints A for exam grades greater than or equal to 90, B for grades in the range 80 to 89, C for grades in the range 70 to 79, D for grades in the range 60 to 69 and F for all other grades.

- Nested `if...else` statements test for multiple cases by placing `if...else` selection statements inside other `if...else` selection statements.

```
If student's grade is greater than or equal to 90
    Print "A"
Else
    If student's grade is greater than or equal to 80
        Print "B"
    Else
        If student's grade is greater than or equal to 70
            Print "C"
        Else
            If student's grade is greater than or equal to 60
                Print "D"
            Else
                Print "F"
```

Nested if .. else statements

```
if ( studentGrade >= 90 ) // 90 and above gets "A"
   cout << "A";
else
   if ( studentGrade >= 80 ) // 80-89 gets "B"
      cout << "B";
   else
      if ( studentGrade >= 70 ) // 70-79 gets "C"
         cout << "C";
      else
         if ( studentGrade >= 60 ) // 60-69 gets
//"D"
            cout << "D";
         else // less than 60 gets "F"
            cout << "F";
```

Dangling –else Problem

- Each `else` associated with immediately preceding `if`
- There is exception when placing braces {}
- Example

```
x=10; y=2;
if (x>5)
  if(y>5)
    cout<<“x and y are >5 “<<endl;
  else
    cout<<“x is <=5”;
```

Logical error !!

if...else Double-Selection Statement (cont.)

- Most write the preceding if...else statement as

```
if ( studentGrade >= 90 ) // 90 and above gets "A"
   cout << "A";
else if ( studentGrade >= 80 ) // 80-89 gets "B"
   cout << "B";
else if ( studentGrade >= 70 ) // 70-79 gets "C"
   cout << "C";
else if ( studentGrade >= 60 ) // 60-69 gets "D"
   cout << "D";
else // less than 60 gets "F"
   cout << "F";
```

Dangling-else Problem

- Correctness

```
x=10; y=2;
if(x>5)
  {
    if(y>5)
    cout<<“x and y are >5”<endl;
  }
else
  cout<<“x is <=5”;
```

if...else Double-Selection Statement (cont.)

- The C++ compiler always associates an `else` with the immediately preceding `if` unless told to do otherwise by the placement of braces (`{` and `}`).
- This behavior can lead to what's referred to as the dangling-`else` problem.
- ```
  x=10; y=2;
  ```
  - ```
    if ( x > 5 )
       if ( y > 5 )
          cout << "x and y are > 5";
    else
       cout << "x is <= 5";
    ```

  appears to indicate that if `x` is greater than `5`, the nested `if` statement determines whether `y` is also greater than `5`.

if...else Double-Selection Statement (cont.)

- The compiler actually interprets the statement as

```
x=10; y=2;
if ( x > 5 )
   if ( y > 5 )
      cout << "x and y are > 5";
   else
      cout << "x is <= 5";
```

- To force the nested if…else statement to execute as intended, use:

```
x=10; y=2;
if ( x > 5 )
{
   if ( y > 5 )
      cout << "x and y are > 5";
}
else
   cout << "x is <= 5";
```

- Braces ({}) indicate that the second if statement is in the body of the first if and that the else is associated with the first if.

Example:

```
if( n > 0 )
{  if( n%2 == 1 )
     cout << " Positive odd number \n";
}
else
  cout << " Negative number or zero\n";
```

Using Boolean variables

```
bool  flag1, flag2;

  if (flag1)
    ----
  else
    ----


  if( flag1|| flag2)
      ----
  else
      -----
```

Implicit Typecasting

```
int x1,x2;

if(x1)
  …
else
  …


if(x1||x2)
  …
else
  ….
```

Note

- Confusing the equality operator == with the assignment operator = results in logic errors.

```
#include <iostream>
using namespace std;
int main ()
{
        int num=0,x=0;

        if (x=2)
        cout<<"x is equal to 2";
        else
        cout<<"x is not equal to 2";

        return 0;
}
```

This message will always be printed !!

Example:

```
// speed.cpp
// Output the fine for driving too fast.

#include <iostream>
using namespace std;

int main()
{
   float limit, speed, toofast;
   cout << "\nSpeed limit: ";
   cin >> limit;
   cout << "\nSpeed: ";
   cin >> speed;

   if( (toofast = speed - limit ) < 10)
     cout << "You were lucky!" << endl;
   else if( toofast < 20)
     cout << "Fine payable: 40,-. Dollars" << endl;
   else if( toofast < 30)
     cout << "Fine payable: 80,-. Dollars" << endl;
   else
     cout << "Hand over your driver's license!" << endl;
   return 0;
}
```

Example:

```
// if_else.cpp
// Demonstrates the use of if-else statements

#include <iostream>
using namespace std;
int main()
{
   float x, y, min;

   cout << "Enter two different numbers:\n";
   if( cin >> x && cin >> y)   // If both inputs are
   {                           // valid, compute
     if( x < y )               // the lesser.
        min = x;
     else
       min = y;
     cout << "\nThe smaller number is: " << min << endl;
   }
   else
     cout << "\nInvalid Input!" << endl;

   return 0;
}
```

Example

- State the output for each of the following:
  - When x is 9 and y is 11
  - When x is 11 and y is 9

a.

```
if(x<10)
if(y>10)
cout<<“*****”<<endl;
else
cout<“#####”<<endl;
cout<<“$$$$$”<<endl;
```

Example:

b.

```
if(x<10)
{
if(y>10)
cout<<"*****"<<endl;
}
else
{
cout<<"#####"<<endl;
cout<<"$$$$$"<<endl;
}
```

Example:

```
// greater.cpp
#include <iostream>
using namespace std;

int main()
{
    float x, y;

    cout << "Type two different numbers:\n";
    if( !(cin >> x && cin >> y) ) // If the input was
    {                            // invalid.
       cout << "\nInvalid input!" << endl;
    }
    else
    {
       cout << "\nThe greater value is: "
            << (x > y ? x : y)  << endl;
    }

    return 0;
}
```

Sample output for this program

```
Type two different numbers:
173.2
216.7
The greater value is: 216.7
```

Example:

- Write a C++ program that compares two integers using if statements, relational operators and equality operators.
- Sample output:

```
Enter two integers to compare:
3
7
3!=7
3<7
3<=7
```

Switch Multiple-selection Statement

- Perform actions based on possible values of variable or expression
- Use keywords `switch`, `case`, `default` and `break`.
- Begin with `switch` followed by controlling expression.
- Value of expression compared to `case` label then go to execute action for that case.
- No matching, the execution go to the optional `default` statement.
- `Break` causes immediate exit from `switch` statement.

Switch Multiple-selection Statement

```
switch (expression)
{
case value1:
     action1;
     break;
case value2:
     action2;
     break;
….
case valuen:
     actionN;
     break;
default:
     action;
}
```

Switch Multiple-selection Statement

- Example:

```
switch (number)
{
  case 0: cout<<“too small, sorry!”;
  break;
  case 5: cout<<“good  job!”<<endl; //fall through
  case 4: cout<<“nice pick!”<<endl; //fall through
  case 3: cout<<“excellent !”<<endl; //fall through
  case 2: cout<<“masterfull!”<<endl; //fall through
  case 1: cout<<“incredible!”<<endl; //fall through
  break;
}
```

Switch Examples:

```
#include <iostream>
using namespace std;
int main(){
char choice=‘’;
cout<<“ please, enter your choice\n”;
cin>>choice;
switch(choice)
    {
        case ‘a’ :
        case ‘A’:
        cout<<“ your choice is A or a”;
        break;
        case ‘b’:
        case ‘B’:
        cout<<“ your choic is B or b’;
        break;
        case ‘c’:
        case ‘C’:
        cout<< “your choice is C or c”;
        break;
        defult:
        cout<<“ your choice is not defined”;
    }
return 0;
}
```

Switch Examples:

- The **break** keyword means "jump out of the switch statement, and do not execute any more code." To show how this works, examine the following piece of code:

```
int value = 0;
switch(input){
  case 1:
    value+=4;
    break;
  case 2:
    value+=3;
    break;
  case 3:
    value+=2;
    break;
  default:
    value++;
}
```