

C++ Programming Essentials

Unit 2: Structured Programming

CHAPTER 5: DECISIONS

DR. ERIC CHOU

IEEE SENIOR MEMBER

LECTURE 1

Program Control Flow



Chapter 5 Topics

- Data Type bool
- Using Relational and Logical Operators to Construct and Evaluate Logical Expressions
- *If-Then-Else* Statements
- *If-Then* Statements
- Nested If Statements for Multi-way Branching
- Switch Statement for Multi-Way Branching
- Testing the State of an I/O Stream
- Testing a C++ Program



Flow of Control

the order in which program statements are executed

WHAT ARE THE POSSIBILITIES. . .



Flow of Control

- is **Sequential** unless a “control structure” is used to change that
- there are 2 general types of control structures:
 - **Selection** (also called branching)
 - **Repetition** (also called looping)

LECTURE 2

Logical Expression



bool Data Type

- type **bool** is a built-in type consisting of just 2 values, the constants **true** and **false**
- we can declare variables of type bool

```
bool  hasFever;    // true if has high temperature  
bool  isSenior;    // true if age is at least 55
```



C++ control structures

- Selection
 - if
 - if . . . else
 - switch
- Repetition
 - for loop
 - while loop
 - do . . . while loop



Control Structures

- use logical expressions which may include:

6 Relational Operators

< <= > >= == !=

3 Logical Operators

! && ||



6 Relational Operators

are used in expressions of form:

<i>ExpressionA</i>	<i>Operator</i>	<i>ExpressionB</i>
--------------------	-----------------	--------------------

temperature > humidity

B * B - 4.0 * A * C > 0.0

abs (number) == 35

initial != 'Q'

```
int x, y ;
```

```
x = 4;
```

```
y = 6;
```

<u>EXPRESSION</u>	<u>VALUE</u>
$x < y$	true
$x + 2 < y$	false
$x \neq y$	true
$x + 3 \geq y$	true
$y == x$	false
$y == x + 2$	true
$y = x + 3$	7 (true)



In C++

- the value 0 represents false
- ANY non-zero value represents true
 - By default true is represented as a 1



Comparing Strings

- two objects of type string (or a string object and a C string) can be compared using the relational operators
- a character-by-character comparison is made using the ASCII character set values
- if all the characters are equal, then the 2 strings are equal. Otherwise, the string with the character with smaller ASCII value is the “lesser” string

```
string myState;  
string yourState;  
  
myState = "Texas";  
yourState = "Maryland";
```

<u>EXPRESSION</u>	<u>VALUE</u>
myState == yourState	false
myState > yourState	true
myState == "Texas"	true
myState < "texas"	true

LECTURE 3

Boolean Operators

Operator	Meaning	Associativity
!	NOT	Right
*, / , %	Multiplication, Division, Modulus	Left
+, -	Addition, Subtraction	Left
<	Less than	Left
<=	Less than or equal to	Left
>	Greater than	Left
>=	Greater than or equal to	Left
==	Is equal to	Left
!=	Is not equal to	Left
&&	AND	Left
 	OR	Left
=	Assignment	Right

LOGICAL EXPRESSION	MEANING	DESCRIPTION
! p	NOT p	! p is false if p is true ! p is true if p is false
p && q	p AND q	p && q is true if both p and q are true. It is false otherwise.
p q	p OR q	p q is true if either p or q or both are true. It is false otherwise.

```
int    age ;  
bool   isSenior, hasFever ;  
float  temperature ;  
  
age = 20;  
temperature = 102.0 ;  
isSenior = (age >= 55) ;           // isSenior is false  
hasFever = (temperature > 98.6) ; // hasFever is true
```

<u>EXPRESSION</u>	<u>VALUE</u>
isSenior && hasFever	false
isSenior hasFever	true
! isSenior	true
! hasFever	false



What is the value?

```
int age, height;  
age = 25;  
height = 70;
```

EXPRESSION	VALUE
!(age < 10)	?
!(height > 60)	?

LECTURE 4

Short-Circuit Evaluation



“Short-Circuit” Evaluation

- C++ uses short circuit evaluation of logical expressions
- this means logical expressions are evaluated left to right and evaluation stops as soon as the final truth value can be determined



Short-Circuit Example

```
int age, height;  
age = 25;  
height = 70;
```

EXPRESSION

(age > 50) && (height > 60)

false

- Evaluation can stop now because result of && is only true when both sides are true. It is already determined that the entire expression will be false.



More Short-Circuiting

```
int age, height;  
age = 25;  
height = 70;
```

EXPRESSION

(height > 60) || (age > 40)
true

- Evaluation can stop now because result of || is true if one side is true. It is already determined that the entire expression will be true.

What happens?

```
int age, weight;  
age = 25;  
weight = 145;
```

EXPRESSION

(weight < 180) && (age >= 20)

true

Must still be evaluated because truth value of entire expression is not yet known. Why? Result of && is only true if both sides are true.

What happens?

```
int age, height;  
age = 25;  
height = 70;
```

EXPRESSION

!(height > 60) || (age > 50)

true

false

Does this part need to be evaluated?

LECTURE 5

Formulate a Logic Expression



Write an expression for each

- taxRate is over 25% and income is less than \$20000
- temperature is less than or equal to 75 or humidity is less than 70%
- age is over 21 and age is less than 60
- age is 21 or 22



Some Answers

`(taxRate > .25) && (income < 20000)`

`(temperature <= 75) || (humidity < .70)`

`(age > 21) && (age < 60)`

`(age == 21) || (age == 22)`



Use Precedence Chart

int number ;

float x ;

number != 0 && x < 1 / number

/ has highest priority

< next priority

!= next priority

&& next priority

What happens if Number has value 0?

Run Time Error (Division by zero) occurs.



Short-Circuit Benefits

- one Boolean expression can be placed first to “guard” a potentially unsafe operation in a second Boolean expression
- time is saved in evaluation of complex expressions using operators `||` and `&&`



Our Example Revisited

```
int  number;
```

```
float x;
```

```
( number != 0) && ( x < 1 / number )
```

is evaluated first and has value false

- Because operator is &&, the entire expression will have value false. Due to short-circuiting the right side is not evaluated in C++.



WARNING about Expressions in C++

- “Boolean expression” means an expression whose value is true or false
- an expression is any valid combination of operators and operands
- each expression has a value
- this can lead to UNEXPECTED RESULTS
- construct your expressions CAREFULLY
- use of parentheses is encouraged
- otherwise, use precedence chart to determine order



What went wrong?

- This is only supposed to display “HEALTHY AIR” if the air quality index is between 50 and 80.
- But when you tested it, it displayed “HEALTHY AIR” when the index was 35.

```
int AQIndex ;  
AQIndex = 35 ;  
if (50 < AQIndex < 80)  
    cout << “HEALTHY AIR” ;
```



Analysis of Situation

- $AQIndex = 35;$
- According to the precedence chart, the expression
- $(50 < AQIndex < 80)$ *means*
- $(50 < AQIndex) < 80$ *because $<$ is Left Associative*
- $(50 < AQIndex)$ is false *(has value 0)*
- $(0 < 80)$ is true.



Corrected Version

```
int AQIndex ;  
AQIndex = 35 ;  
  
if ( (50 < AQIndex) && (AQIndex < 80) )  
  
    cout << "HEALTHY AIR" ;
```



Comparing float Values

- do not compare float values for equality, compare them for **near-equality**.

```
float myNumber;  
float yourNumber;  
  
cin >> myNumber;  
cin >> yourNumber;  
  
if ( fabs (myNumber - yourNumber) < 0.00001 )  
    cout << "They are close enough!" << endl;
```

LECTURE 6

Boolean Expression in other Control Structure



Flow of Control

the order in which program statements are executed

THE 3 POSSIBILITIES ARE:

Sequential

Selection Control Structure

Loop Control Structure



In C++

- the value 0 represents false
- ANY non-zero value represents true
- `#include <cstdbool>` `// stdbool.h in C language`



What can go wrong here?

```
float average;  
float total;  
int  howMany;  
  
.  
.  
.  
  
average = total / howMany;
```


Improved Version

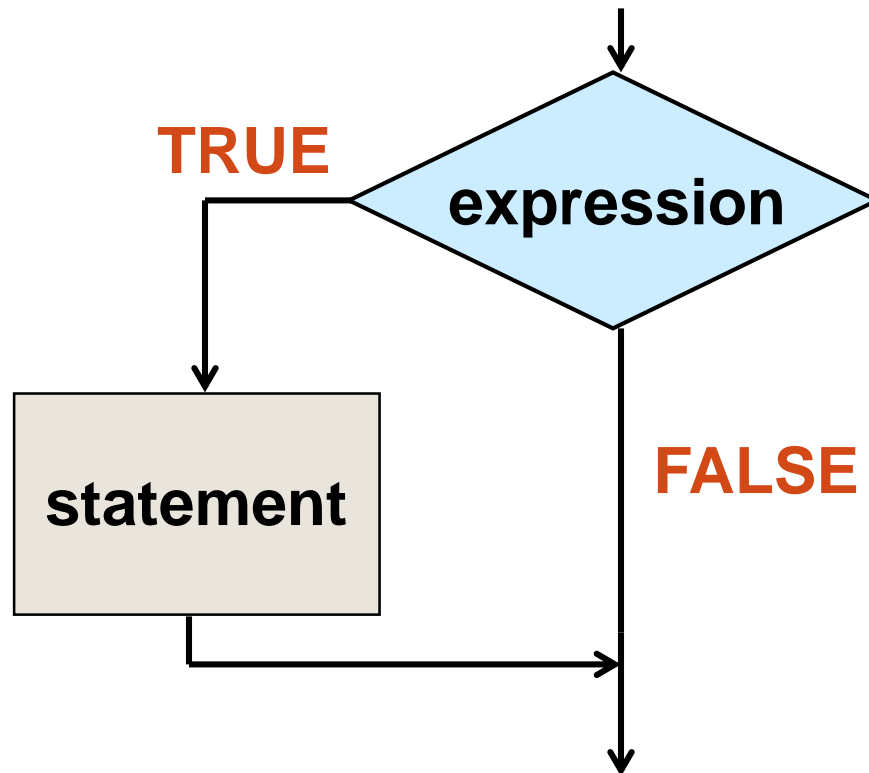
```
float average,  
float total;  
int  howMany;  
  
if ( howMany > 0 )  
{  
    average = total / howMany;  
    cout << average;  
}  
else  
    cout << "No prices were entered";
```

LECTURE 7

If-then and if-then-else Statement

If-Then statement is a selection

- of whether or not to execute a statement (which can be a single statement or an entire block)





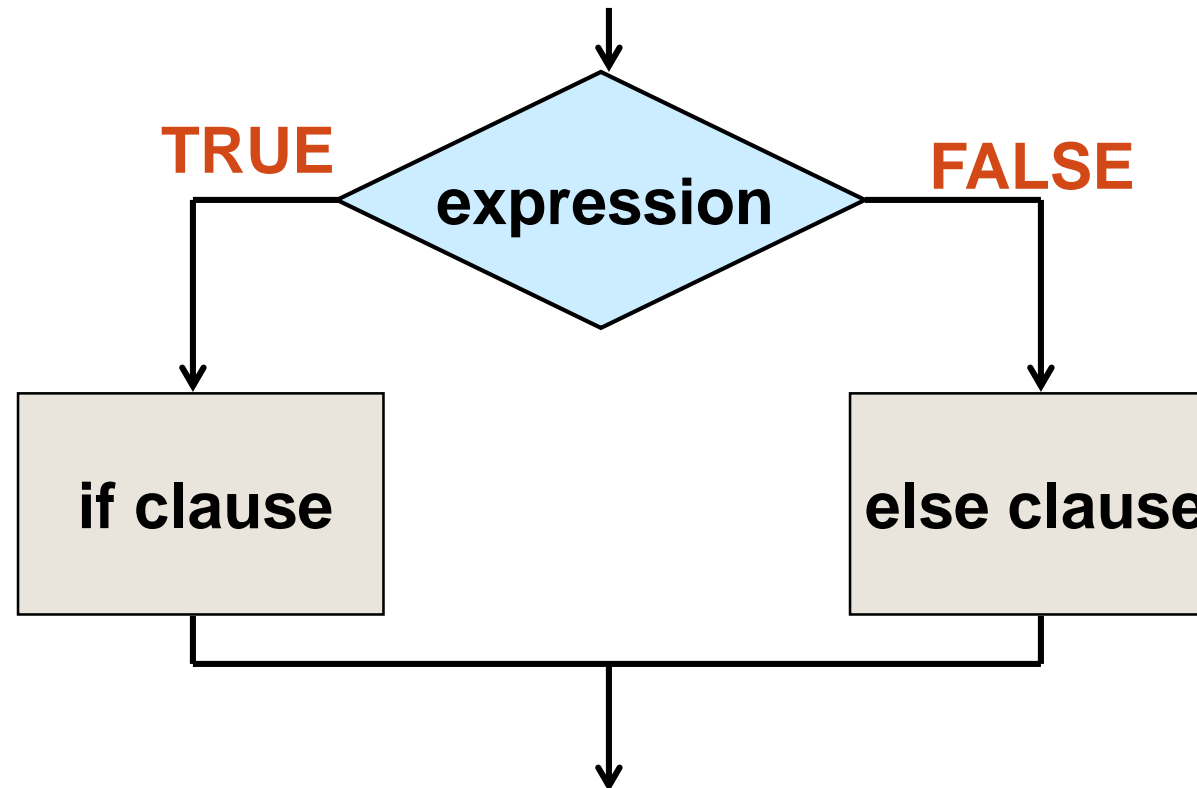
If-Else Syntax

```
if ( Expression )  
  
    Statement
```

NOTE: Statement can be a single statement, a null statement, or a block.

if ... else provides two-way selection

between executing one of 2 clauses (the if clause or the else clause)





If-Then-Else Syntax

```
if ( Expression )  
    StatementA  
else  
    StatementB
```

NOTE: StatementA and StatementB each can be a single statement, a null statement, or a block.

Use of blocks recommended

```
if ( Expression ){  
    }  
else{  
    }  
}
```

Diagram illustrating the use of blocks in C++ code:

- The `if (Expression)` line is connected by a red arrow to the text **"if clause"**.
- The `else{` line is connected by a red arrow to the text **"else clause"**.

```
int    carDoors, driverAge ;  
    float premium, monthlyPayment ;  
    . . .  
if ( (carDoors == 4 ) && (driverAge > 24) ){  
    premium = 650.00 ;  
    cout << " LOW RISK " ;  
}  
else{  
    premium = 1200.00 ;  
    cout << " HIGH RISK " ;  
}  
monthlyPayment = premium / 12.0 + 5.00 ;
```




What happens if you omit braces?

```
if ( (carDoors == 4 ) && (driverAge > 24) )  
    premium = 650.00 ;  
    cout << " LOW RISK " ;  
  
else  
    premium = 1200.00 ;  
    cout << " HIGH RISK " ;  
  
monthlyPayment = premium / 12.0 + 5.00 ;
```

COMPILE ERROR OCCURS.

The “if clause” is the single statement following the if.



Braces can only be omitted when each clause is a single statement

```
if ( lastInitial <= 'K' )  
    volume = 1;  
else  
    volume = 2;  
  
cout << "Look it up in volume # "  
      << volume << " of NYC phone book";
```



Determining where the first 'A' was found in a string

```
string myString ;  
string::size_type pos;  
  
. . .  
pos = myString.find('A');  
if ( pos == string::npos )  
    cout << "No 'A' was found" << endl ;  
else  
    cout << "An 'A' was found in position " << pos << endl ;
```

LECTURE 8

Demo Program: Mail Order



If-Then-Else for a mail order

- Assign value .25 to discountRate and assign value 10.00 to shipCost if purchase is over 100.00
- Otherwise, assign value .15 to discountRate and assign value 5.00 to shipCost
- Either way, calculate totalBill



These braces cannot be omitted

```
if ( purchase > 100.00 )
{
    discountRate = .25 ;
    shipCost = 10.00 ;
}
else
{
    discountRate = .15 ;
    shipCost = 5.00 ;
}
totalBill = purchase * (1.0 - discountRate) + shipCost ;
```



Terminating your program

```
int number ;  
cout << "Enter a non-zero number " ;  
cin  >> number ;  
if (number == 0 ) {  
    cout << "Bad input. Program terminated " ;  
    return 1 ; // exit with the error code improper operation  
}  
  
// otherwise continue processing
```



These are equivalent. Why?

```
if (number == 0 )
```

```
{ .
```

```
 .
```

```
 .
```

```
 .
```

```
}
```

```
if ( ! number )
```

```
{ .
```

```
 .
```

```
 .
```

```
 .
```

```
}
```

Each expression is only true when number has value 0.

LECTURE 9

Application of If-then and If-then-else Statements



Write *If-Then* or *If-Then-Else* for each

- If taxCode is 'T', increase price by adding taxRate times price to it.
- If code has value 1, read values for income and taxRate from myInfile, and calculate and display taxDue as their product.
- If A is strictly between 0 and 5, set B equal to $1/A$, otherwise set B equal to A.



Some Answers

```
if (taxCode == 'T')
```

```
    price = price + taxRate * price;
```

```
if ( code == 1)
```

```
{
```

```
    myInfile >> income >> taxRate;
```

```
    taxDue = income * taxRate;
```

```
    cout << taxDue;
```

```
}
```



Remaining Answer

```
if ( ( A > 0 ) && ( A < 5 ) )
```

```
    B = 1/A;
```

```
else
```

```
    B = A;
```



What output? and Why?

```
int age;  
age = 20;  
if ( age = 16 ) {  
    cout << "Did you get driver's license?" ;  
}
```



What output? and Why?

```
int age;  
age = 30;  
if ( age < 18 )  
  
    cout << "Do you drive?";  
    cout << "Too young to vote";
```



What output? and Why?

```
int code;  
code = 0;  
if ( ! code )  
    cout << "Yesterday";  
else  
    cout << "Tomorrow";
```



What output? and Why?

```
int number;  
number = 0;  
if ( number = 0 )  
    cout << "Zero value";  
else  
    cout << "Non-zero value";
```




Both the if clause and the else clause

of an if...else statement can contain any kind of statement, including another selection statement.

LECTURE 10

Multiple-Alternative Selection and Nested If



Multi-alternative Selection

is also called **multi-way branching**, and can be accomplished by using NESTED if statements.

Nested if Statements

```
if ( Expression1 )  
    Statement1  
  
else if ( Expression2 )  
    Statement2  
    .  
    .  
    .  
else if ( ExpressionN )  
    StatementN  
else  
    Statement N+1
```

EXACTLY 1 of these statements will be executed.



Nested if Statements

- Each Expression is evaluated in sequence, until some Expression is found that is true.
- Only the specific Statement following that particular true Expression is executed.
- If no Expression is true, the Statement following the final else is executed.
- Actually, the final else and final Statement are optional. If omitted, and no Expression is true, then no Statement is executed.

AN EXAMPLE . . .

Multi-way Branching

```
if ( creditsEarned >= 90 )  
    cout << "SENIOR STATUS ";  
else if ( creditsEarned >= 60 )  
    cout << "JUNIOR STATUS ";  
else if ( creditsEarned >= 30 )  
    cout << "SOPHOMORE STATUS ";  
else  
    cout << "FRESHMAN STATUS ";
```



Writing Nested if Statements

- Display one word to describe the int value of number as “Positive”, “Negative”, or “Zero”
- Your city classifies a pollution index
 - less than 35 as “Pleasant”,
 - 35 through 60 as “Unpleasant”,
 - and above 60 as “Health Hazard.”
 - Display the correct description of the
 - pollution index value.



One Answer

```
if (number > 0)
    cout << "Positive";
else if (number < 0)
    cout << "Negative";
else
    cout << "Zero";
```




Other Answer

```
if ( index < 35 )  
    cout << "Pleasant";  
else if ( index <= 60 )  
    cout << "Unpleasant";  
else  
    cout << "Health Hazard";
```

LECTURE 12

Using Function with Selection to Handle I/O



Write a void Function

- called `DisplayMessage` which you can call from main to describe the pollution index value it receives as an argument.
- Your city describes a pollution index
less than 35 as “Pleasant”,
35 through 60 as “Unpleasant”,
and above 60 as “Health Hazard.”

```
void DisplayMessage( int index )
{
    if ( index < 35 )

        cout << "Pleasant";

    else if ( index <= 60 )

        cout << "Unpleasant";

    else

        cout << "Health Hazard";

}
```

The Driver Program

```
#include <iostream>

using namespace std;

void DisplayMessage (int);  // declare function

int main (void)
{
    int pollutionIndex;      // declare variable


    cout << "Enter air pollution index:";
    cin >> pollutionIndex;
    DisplayMessage (pollutionIndex);  // call
    return 0;
}
```



Demo Program:

`displaymessage2.cpp`

Go Dev C++!!!

 C:\Eric_Chou\Cpp Course\C++ Programming Essentials\CppDev\ch5\DisplayMessage2\displaymessage2.exe

```
Enter air pollution index: 4  
Pleasant
```



Using selection

- Every Monday thru Friday you go to class.
 - When it is raining you take an umbrella.
- But on the weekend, what you do depends on the weather.
 - If it is raining you read in bed. Otherwise, you have fun outdoors.

```
// program tells how to spend your day
#include < iostream >
using namespace std;

void main ( void )
{
    int    day;
    char   raining;

    cout << "Enter day (use 1 for Sunday):";
    cin  >> day;

    cout << "Is it raining? (Y/N)";
    cin  >> raining;
```



```
if ( ( day == 1) || (day == 7) )    // Sat or Sun
{
    if (raining == 'Y')
        cout << "Read in bed";
    else
        cout << "Have fun outdoors";
}
else
{
    cout << "Go to class ";
    if (raining == 'Y')
        cout << "Take an umbrella";
}
}
```

**In the absence of braces,
an else is always paired
with the closest preceding if that doesn't
already have an else paired with it.**

Bad Example has output: FAIL

```
float average;
```

```
average = 100.0;
```

```
if ( average >= 60.0 )
```

```
    if ( average < 70.0 )
```

```
        cout << "Marginal PASS";
```

```
else
```

```
    cout << "FAIL";
```

100.0

average

WHY? The compiler ignores indentation and pairs the else with the second if.

To correct the problem, use braces

```
float average;  
  
average = 100.0;  
  
if ( average >= 60.0 )  
{  
    if ( average < 70.0 )  
        cout << "Marginal PASS";  
}  
else  
    cout << "FAIL";
```

100.0

average

LECTURE 13

Checking I/O States



Each I/O stream has a state (condition)

- An **input stream** enters fail state when you
 - try to read invalid input data
 - try to open a file which does not exist
 - try to read beyond the end of the file
- An output stream enters fail state when you
 - try to create a file with an invalid name
 - try to create a file on a write-protected disk
 - try to create a file on a full disk



How can you tell the state?

- The stream identifier can be used as if it were a Boolean variable. It has value false (meaning the last I/O operation on that stream failed) when the stream is in fail state.
- When you use a file stream, you should check on its state.

Checking on the State

```
ofstream myOutfile;  
  
myOutfile.open ("A:\\myOut.dat");  
  
if ( ! myOutfile )  
{  
    cout << "File opening error. "  
        << "Program terminated." << endl;  
    return 1;  
}  
  
// otherwise send output to myOutfile
```


LECTURE 14

Switch Statement



Switch Statement

- The Switch statement is a selection control structure for multi-way branching

```
switch (IntegralExpression){  
    case Constant1 :  
        Statement(s);           // optional  
    case Constant2 :  
        Statement(s);           // optional  
        .  
        .  
        .  
    default :                     // optional  
        Statement(s);           // optional  
}
```



Example of Switch Statement

```
float    weightInPounds    = 165.8;
char     weightUnit;

. . . // User enters letter for desired weightUnit

switch (weightUnit){
    case 'P' :
    case 'p' :
        cout << weightInPounds << " pounds " << endl;
    break;
    case 'O' :
    case 'o' :
        cout << 16.0 * weightInPounds << " ounces
" << endl;
```



Example of Switch Statement, continued

```
break;
    case 'G' :
    case 'g' :
        cout << 454.0 * weightInPounds
              << " grams " << endl;
        break;
    default :
        cout << "That unit is not handled! "
              << endl;
        break;
}
```



Switch Statement

- The value of **IntegralExpression** (of char, short, int, long or enum type) determines which branch is executed
- Case labels are constant (possibly named) integral expressions
- Several case labels can precede a statement



Control in Switch Statement

- Control branches to the statement following the case label that matches the value of **IntegralExpression**
- Control proceeds through all remaining statements, including the default, unless redirected with break



Control in Switch Statement

- If no case label matches the value of **IntegralExpression**, control branches to the default label, if present
- Otherwise control passes to the statement following the entire switch statement
- **Forgetting to use break can cause logical errors** because after a branch is taken, control proceeds sequentially until either break or the end of the switch statement occurs

LECTURE 15

Selection Control Structure for Program Testing



Testing Selection Control Structures

- to test a program with branches, use enough data sets so that every branch is executed at least once
- this is called **minimum complete coverage**



Testing Often Combines Two Approaches

WHITE BOX TESTING

Code Coverage

Allows us to see the program code while designing the tests, so that data values at the boundaries, and possibly middle values, can be tested.

BLACK BOX TESTING

Data Coverage

Tries to test as many allowable data values as possible without regard to program code.



How to Test a Program

- design and implement a test plan
- a **test plan** is a document that specifies the test cases to try, the reason for each, and the expected output
- implement the test plan by verifying that the program outputs the predicted results



PHASE

RESULT

TESTING TECHNIQUE

Problem solving	Algorithm	Algorithm walk-through
Implementation	Coded program Trace	Code walk-through,
Compilation	Object program	Compiler messages
Execution	Output	Implement test plan