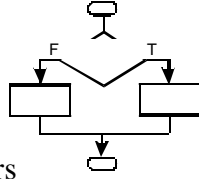


Topic: branch statements

- *if*
- *if ... else*
- Compound statements
- Relational operators
- Boolean variables
- Logical operators
- *switch*



David Keil 9/03 1

Using *if* to validate input

```

void main() [age1.cpp]
{
  cout << "Enter your age: ";
  int age;
  cin >> age;
  if (age < 0)
    cout << "Invalid age\n";
}
  
```

- An *if* statement consists of the keyword *if*, a Boolean expression (condition) in parentheses, and *one* statement that is executed if the condition is true

David Keil 9/03 2

if with compound statement

```

int age;
cin >> age;
if (age < 0)
{
  cout << "Invalid age";
  cin >> age;
}
  
```

Omitting braces here would change meaning!

- When multiple statements execute conditional on an *if*, they must be part of a compound statement

David Keil 9/03 3

Compound statements and scope of access

```

void main() [expdate.cpp]
{
  int curr_year = 1998, expir_date = 2000;
  if (curr_year <= expir_date)
  {
    cout << "Amount of purchase: ";
    double amount = 0.0;
    cin >> amount;
  }
  cout << amount;
}
  
```

- The scope of access of an identifier is the *block* (compound statement) where it is declared
- For code above to work, put *cout* within braces

David Keil 9/03 4

if...else provides two paths of execution

```

char line[80];
ifstream
infile("X.TXT", ios::nocreate);
if (infile.good())
  infile.getline(line, 79);
else
  cout << "X.TXT not found." << endl;
  
```

[readline.cpp]

Tip: Check any file-opening operation before proceeding

David Keil 9/03 5

else saves the result of a test

```

cout << "How many CDs? ";
int quantity;
cin >> quantity;
if (quantity >= 5)
  cout << "Discount 10%\n";
if (quantity < 5)
  cout << "No discount\n";
  
```

Equivalent branch statement:

```

if (quantity >= 5)
  cout << "Discount 10%\n";
else
  cout << "No discount\n";
  
```

[discount.cpp]

David Keil 9/03 6

What is the output, on inputs $a = 3, b = 5$?

```
void main()
{
    int a,b;
    cin >> a >> b;
    if (a > 2)
        if (b < 4)
            cout << "OK";
    else
        cout << "a <= 2";
    cout << "Done";
}
```

- The code above has a logic error. What is it?

The pairing rule with *else*:

Each *else* is paired with the most recent unpaired *if* in the same scope.

Example (misleading code):

```
if (test1)
    if (test2)
        statement1();
else
    statement2();
```

(If *test2* fails, *statement2* executes.)

Relational operators

equal-to	==	not-equal-to	!=
greater	>	less-than-or-equal	<=
less-than	<	greater-or-equal	>=

- Each operator has a complement
- Tip: don't compare *floats* or *doubles* for equality
- Can't compare C-style strings with any relational operators; use *strcmp*

Standard functions and *if*

```
char input[20];
cin >> input; // declared in ctype.h
if (isdigit(input[0]))
    cout << "Input starts with digit";
```

```
char name[20];
cin >> name; // declared in string.h
if (strcmp(name,"M") > 0)
    cout << "Name is in M..Z";
```

```
...
if (! strcmp(command,"quit"))
    cout << "Command is 'quit'";
```

== is a relational operator; = is not

```
void main()
{
    cout << "Enter your age: ";
    int age;
    cin >> age; // Watch out!
    if (age = 0)
        cout << "Invalid\n";
}
```

[agebad.cpp]

- Assignment as an *if* condition is valid syntax but usually logic error
- With assignment of constant, in effect no test is performed

A Boolean variable (flag) stores a truth value

may be replaced by int a flag

```
• bool invalid = (age < 0);
...
if (invalid)
    cout << "Invalid age" << endl;
```

- bool* is a standard ANSI/ISO C++ type with a range of values {*false, true*} (0, 1)
- Boolean variables hold values for later use

Logical operators form Boolean expressions

Operation	ANSI C++	C	Example
Negation	not	!	!(price > cost)
Conjunction	and	&&	a > b && b > c
Disjunction	or	 	x == 1 x == 2

- Nested *ifs* may express conjunction too:


```
if (age > 0)
    if (age < 120)
        cout << "Valid age";
```
- The above is equivalent to


```
if (age > 0 && age < 120)
    cout << "Valid age";
```

David Keil 9/03 13

Using logical operators

```
void main()
{
    cout << "Enter 3 integers: ";
    int a, b, c;
    cin >> a >> b >> c;
    if (a == b && b == c)
        cout << "They're the same!" << endl;

    cout << "Enter your age: ";
    int age;
    cin >> age;
    bool impossible = (age < 0 || age > 120);
    if (!impossible)
        cout << "Thank you" << endl;
}
```

[logops.cpp] David Keil 9/03 14

Truth tables

a	!a
1	0
0	1

a	b	a b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a&& b
0	0	0
0	1	0
1	0	0
1	1	1

- The logical operators in C/C++ have the same results as the basic logic gates used in computer hardware

David Keil 9/03 15

Exercise all paths in testing a module

- To test a program containing an *if* statement, you must use test data that will cause a *true* value and a *false* value to appear as the result of the *if* condition
- With nested *ifs*, fully exercising your code (getting test results that reflect execution of each statement in program) may require 4 tests, 8, etc.

David Keil 9/03 16

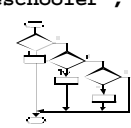
Multiway branches

Nested if:

```
if (age == 1) cout << "Baby";
else if (age == 2) cout << "Toddler";
else if (age == 3) cout << "Preschooler";
else cout << "Other";
```

switch:

```
switch (age)
{
    case 1: cout << "Baby"; break;
    case 2: cout << "Toddler"; break;
    case 3: cout << "Preschooler"; break;
    default: cout << "Other";
}
```



David Keil 9/03 17

The switch statement: guidelines

- Appropriate for mutually exclusive cases
- Test for exact matches, not ranges of values
- Selector should be an *int* or *char*
- Case labels may be stacked
- Don't forget to use *break!*
- default* triggers processing when no *case* label matches the selector

David Keil 9/03 18

Stacking case labels

```

#include <ctype.h>
void main()
{
    cout << "Grade: ";
    char grade;
    cin >> grade;
    switch(toupper(grade))
    {
        case 'A':
        case 'B':
            cout << "Honors" << endl;    break;
        case 'C':
        case 'D':
            cout << "Passing" << endl;  break;
        case 'E':
            cout << "Failing" << endl;  break;
        default:
            cout << "Invalid input" << endl;
    }
}
    
```

Stacked to accept either 'A' or 'B'

David Keil 9/03 19

Syntax for branches

branch-statement:

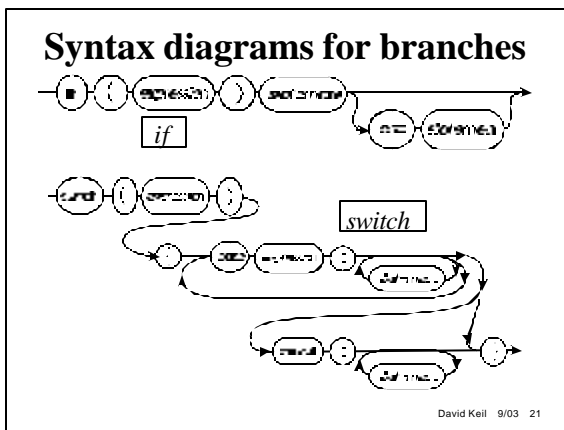
if (*expr*) *statement*

if (*expr*) *statement* **else** *statement*

switch (*expr*) *statement*

The subordinate *statement* in a *switch* statement is normally compound, with case labels, alternative statements, *breaks*, and a single optional *default* label.

David Keil 9/03 20



Discussion problems

- Write a program that accepts two floating-point numbers and displays the larger one.
- Write a program that accepts
 - three numbers and displays the largest;
 - four numbers;
 - five numbers
- Write a program that prompts for two integers and displays their quotient; show an error message if the divisor is 0.
- Write a program that allows the user to input any two real numbers and then choose one of the four operations +, -, *, or /. Display the appropriate computed result. Use a *switch* statement to select an operation.

David Keil 9/03 22