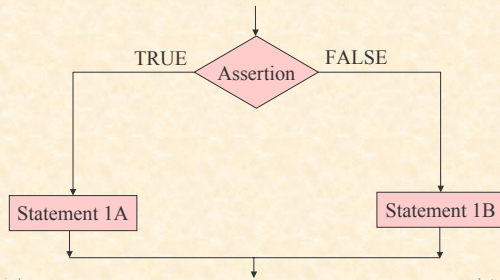


CONTROL STATEMENTS - INTRODUCTION

- Determines flow of control in program
- Allows selection of execution



Programming 1

Lecture 9 -- 1

CONTROL STATEMENTS - EXAMPLE

Algorithm:

1. Read number
2. If number is equal to 4, print "correct"
3. If number is higher then 6, print "too high value"
4. If number is lower then 2, print "too low value"
5. Otherwise print "almost there"

Programming 1

Lecture 9 -- 2

IF-ELSE STATEMENT

- Make a choice based on a **boolean** value
 - if** (something is true) then
 - do action 1
 - else**
 - do action2

Either action 1 or action 2 is executed but not both
- More formally
 - if** (expression)
 - statement 1
 - else**
 - statement 2
 - expression evaluates to TRUE or FALSE
 - note that statements may be single or compound

Programming 1

Lecture 9 -- 3

IF-ELSE STATEMENT

□ Example:

```
#include <iostream.h>
int main()
{
    int const min = 0, max = 100;
    int num = 23;

    if (num >= min && num <= max) {
        cout << num << " is between ";
        cout << min << " and " << max << endl;
    }
    else {
        cout << num << " is not between ";
        cout << min << " and " << max << endl;
    }
    return 0;
}
```

Programming 1

Lecture 9 -- 4

IF-ELSE STATEMENT

- Note that **else** is optional - this can cause ambiguity

```
if (a != b)
    if (a > b)
        cout << "a is greater than b" << endl;
    else
        cout << "a is less than b" << endl;
```

may be confused by **reader** for

```
if (a != b)
    if (a > b)
        cout << "a is greater than b" << endl;
else
    cout << "a is less than b" << endl;
```

- Use compound statement to overcome ambiguity

Programming 1

Lecture 9 -- 5

BOOLEANS

- Standard C++ has boolean type but very few compilers implement it yet
- For now we will consider C++ as not having a boolean type

Instead

- the integer zero is used as **FALSE**
- and non zero value (usually one) is used as **TRUE**

- In an if statement the expression
if (expression)...

must evaluate to an integer

- In if (1 == 2) ... the expression evaluates to **zero**
- In if (2 > 1) ... the expression evaluates to **one**

Programming 1

Lecture 9 -- 6

EXAMPLE TO HIGHLIGHT COMPOUND STATEMENTS

```
#include <iostream.h>
int main ()
{
    int one = 1;
    int two = 2;
    int result;
    if (one == two) {
        // this code is not executed
        cout << "true" << endl;
        int three = one + two;
        result = three;
    }
}
```

Programming 1

Lecture 9 -- 7

EXAMPLE TO HIGHLIGHT COMPOUND STATEMENTS

```
else {
    // this code is executed
    cout << "false" << endl;
    int four = two + two;
    result = four;
}
// note that both three and four are out of scope here
cout << "result = " << result << endl;
return 0;
}
```

Programming 1

Lecture 9 -- 8

CONDITIONAL EXPRESSION

□ C++ provide tertiary operator ?: as an alternative to if-else

- Unlike if-else ?: can be used wherever expression can be used
- Format:

expr1 ? expr2 : expr3

- Read as:

if expr1 evaluates to TRUE then return the result of the evaluation of expr2, else return the result of the evaluation of expr3

if (a > b)

max = a; **is equivalent to** max = (a > b) ? a : b;

else

max = b;

□ Note, given result = (expr) ? float: int;
 result will be of type float

EXAMPLE OF CONDITIONAL EXPRESSION

```
#include <iostream.h>
int main ()
{
    int num1, num2;
    // read in two numbers
    cout << "enter two numbers: ";
    cin >> num1 >> num2;
    if (num1 != num2)
        cout << (num1 > num2) ?
            "the 1st number is larger" :
            "the 2nd number is larger" << endl;
    return 0;
}
```

THE ELSE-IF STATEMENT

□ This is used to nest if statements (for multi-way decisions)

□ Format:

if (expr1)	if (expr1)
statement1	statement1
else if (expr2)	else if (expr2)
statement2	statement2
else if (expr3)	else if (expr3)
statement3	statement3
else	else
statement 4	statement4

□ Note that the **else** is again optional

EXAMPLE OF THE ELSE-IF STATEMENT

```
if (month == 1)
    cout << "January";
else
    if (month == 2)
        cout << "February";       // nested if
    else
        if (month == 3)
            cout << "March";       // nested if
        else
            if (month == 4)
                // nested if
                .
                .
                .
```

THE SWITCH STATEMENT

□ This is used for multi-way statements and is also known as a **case statement**

- Allows selection of one from many choices
- Format:

```
switch (IntegralExpression) {  
    case const_expr1:  
        statements  
    case const_expr2:  
        statements  
    case const_expr3:  
        statements  
    default:  
        statements  
}
```

- Execution will fall through to the next case if a **break** is not forced using a **break** statement.

Programming 1

Lecture 9 -- 13

EXAMPLE OF SWITCH STATEMENT

```
switch (grade) {  
    case '5':  
        case '4': cout << "Good Work";  
                break;  
        case '3': cout << "Average Work";  
                break;  
        case '2':  
        case '1': cout << "Poor Work";  
                NumberInTrouble++;  
                break;  
    default :   cout << grade << " is not a legal grade number.";  
                break;  
}
```

Programming 1

Lecture 9 -- 14

NESTED SWITCHES

```
switch (x) {  
    case 1:  
        switch (y) {  
            case 0:  
                cout << " in y(0) " << endl;   break;  
            case 1:  
                cout << " in y(1) " << endl;   break;  
        }  
        break;  
    case 2:  
        // some more statements  
        break;  
}
```

Programming 1

Lecture 9 -- 15

SWITCH AND IF

- There are a number of differences between **switch** and **if**
- **switch** tests for equality only
 - **if** can evaluate relational or logical expressions
 - no two case constants in the same switch can have identical value

Programming 1

Lecture 9 -- 16

COMPILING OF PROGRAM (NOTES)

- C++ compilation consist of several stages, executed by different programs:
- First, preprocessor parse program files
 - **Preprocessor**, execute first stage of compilation, handles so called "preprocessor directives" like:
 - #include
 - #ifdef
 - #ifndef
 - #define
 - #macro
 - #endif
 - and others
 - Obviously, preprocessor directives starts with # sign.
- Second, compiler itself translate enhanced program files (with preprocessor directive applied)

Programming 1

Lecture 9 -- 17

COMPILING OF PROGRAM (NOTES)

- Third, linker links translated program with libraries to make executable version of program
- Preprocessor directives can be used to control flow of compilation!
 - Not directly execution of program
 - They control translation of program
 - Can be used to separate testing version and final version of program

Programming 1

Lecture 9 -- 18

COMPILING OF PROGRAM (NOTES)

```
□ Example:
#define WITH_OWN_SQRT
#ifdef WITH_OWN_SQRT
    double own_sqrt(double);
#endif

main()
{
    .
    #ifdef WITH_OWN_SQRT
        b = own_sqrt(a);
    #else
        b = sqrt(a);
    #endif
    .
}
```

Programming 1

Lecture 9 -- 19

COMPILING OF PROGRAM (NOTES)

```
#include <iostream.h>
#define DEBUG
#ifdef DEBUG
#define printValue(x) {cout << "\n!!!DEBUG: " << x << " !!!\n";}
#else
#define printValue(x)
#endif

int main()
{
    int a,b,c;
    ....
    printValue(a); printValue(b);printValue(c) ;
    ...
}
```

Programming 1

Lecture 9 -- 20