Straight-line code

- So far all of our programs have followed this basic format:
  - Input some values
  - Do some computations
  - Output the results

- The statements are executed in a sequence, first to last.

Decisions

- Sometimes we want to be able to decide which of two statements to execute:

  ![Decision Diagram]

  - If monthly sales > $3,000 (Y), fee is 2.5%
  - If monthly sales ≤ $3,000 (N), fee is 2.9%

Relational Expressions

- Making decisions require being able to ask “Yes” or “No” questions.
- Relational expressions allow us to do this.
- Relational expressions evaluate to true or false.
- Also called:
  - logical expressions
  - conditional expressions
  - boolean expressions
Relational Expressions

• Boolean literals:
  
  true
  false

  true evaluates to true

• Boolean variables

  bool isPositive = true;
  bool found = false;

  isPositive evaluates to true
  found evaluates to false

4.1 Relational Operators

• Binary operators used to compare expressions:

  <  Less than
  <=  Less than or equal to
  >  Greater than
  >=  Greater than or equal to
  ==  Equals (note: do not use =) !!
  !=  Not Equals

Relational Operators Precedence

• Relational operators are LOWER than arithmetic operators:

  int x, y;
  ...
  x < y - 10 ...  // minus happens first
  ...
  x * 5 >= y + 10 ...  // mult, then plus, then >=

• Relational operators are HIGHER than assignment:

  int x, y;
  ...
  bool t1 = x > 7;       // > then =
  bool t2 = x * 5 >= y + 10;  // *, +, >=, =
4.4 if-else statement

- if-else statement is used to make decisions
  
  ```
  if (expression)
  statement1
  else
  statement2
  ```

- expression is evaluated
  - If it is true, then statement1 is executed. (statement2 is skipped).
  - If it is false, then statement2 is executed. (statement1 is skipped).

if-else example

```cpp
double rate;
double monthlySales;

cout << "Enter monthly sales last month: " ;
cin >> monthlySales;
if (monthlySales > 3000)
  rate = .025;
else
  rate = .029;

double price;
cout << "Enter selling price of item: " ;
cin >> price;
double commission = (price + 3.99) * rate;
cout << "Commission: $" << commission << endl;
```

if (monthlySales > 3000)
rate = .025;
else
rate = .029;

double price;
cout << "Enter selling price of item: " ;
cin >> price;
double commission = (price + 3.99) * rate;
cout << "Commission: $" << commission << endl;

Enter monthly sales last month: 3025
Enter selling price of item: 100
Commission: $2.59975

4.3 The block statement

- a block (or a compound statement) is a set of statements inside braces:

  ```
  {  int x;
      cout << “Enter a value for x: “ << endl;
      cin >> x;
      cout << “Thank you.” << endl;
  }
  ```

- This groups several statements into a single statement.
- This allows us to use multiple statements when by rule only one is allowed.

if (monthlySales > 3000)
rate = .025;
else
rate = .029;

double price;
cout << “Enter selling price of item: “ ;
cin >> price;
double commission = (price + 3.99) * rate;
cout << “Commission: $“ << commission << endl;

Enter monthly sales last month: 3025
Enter selling price of item: 100
Commission: $2.59975
if-else with blocks

- We can use blocks to put more than one statement in the branches of the if-else:

```cpp
int number;
cout << "Enter a number" << endl;
cin >> number;
if (number % 2 == 0)
{
    number = number / 2;
cout << "Even";
}
else
{
    number = (number - 1) / 2;
cout << "Odd";
}
```

4.2 if statement

- The else part of the if-else stmt is optional:

```cpp
if (expression)
statement
```

- expression is evaluated
  - If it is true, then statement is executed.
  - If it is false, then statement is skipped

if statement example

- Example: input validation

```cpp
cout << "Enter a positive number: ";
cin >> x;
if (x <= 0)
{
    cout << "That number is not positive. " << "Please enter a positive number: ";
cin >> x;
}
//do something with x here
```

4.5 Nested if statements

- if-else is a statement. It can occur as a branch of another if-else statement.
Nested if statements

- if-else is a statement. It can occur as a branch of another if-else statement.

```cpp
char bornInUSA;
int age;
cout << "Were you born in the USA (Y/N)?: ";
cin >> bornInUSA;
cout << "Please enter your age: ";
cin >> age;
if (bornInUSA == 'Y')
    if (age >= 35)
        cout << "You qualify to run for President\n";
    else
        cout << "You are too young to run for President\n";
else
    cout << "You must have been born in the US in order "
    << "to run for President" << endl;
```

Common nested if pattern

- Determine letter grade from test score:

```cpp
if (testScore < 60)
    grade = 'F';
else {
    if (testScore < 70)
        grade = 'D';
    else {
        if (testScore < 80)
            grade = 'C';
        else {
            if (testScore < 90)
                grade = 'B';
            else
                grade = 'A';
        }
    }
}
```

4.6 The if-else if Statement

- Not really a different statement, just a different way of indenting the nested if statement from the previous slide:

```cpp
if (testScore < 60)
    grade = 'F';
else if (testScore < 70)
    grade = 'D';
else if (testScore < 80)
    grade = 'C';
else if (testScore < 90)
    grade = 'B';
else
    grade = 'A';
```

- removed braces, put “if (…)” on previous line
- eliminated nested indentation.
4.8 Logical Operators

- Used to create relational expressions from other relational expressions:
  - && AND (binary operator)
    - $a \&\& b$ is true only when both $a$ and $b$ are true
  - || OR (binary operator)
    - $a \| b$ is true whenever either $a$ or $b$ is true
  - ! NOT (unary operator)
    - $!a$ is true when $a$ is false

Logical Operator Precedence

- ! is higher than most operators, so use parentheses:

```cpp
int x;
... !(x < 0 && x > -10) ... // <, >, &&, !
```

- && is higher than ||

```cpp
int x, y;
bool flag;
... flag || x * 5 >= y + 10 && x == 5
// which op is first? second? etc?
```

- && and || are lower than arithmetic+relational operators: parens not usually needed

4.9 Checking Numeric Ranges

- We want to know if $x$ is in the range from 1 to 10 (inclusive)

```cpp
if (1 <= x <= 10) // as in math class
    cout << "YES" << endl;

//WRONG: ((1<=x) <=10) (assume x is -5)
//    => ( false <= 10)
//    => ( 0 < = 10 ) is true
```

```cpp
if (1 <= x && x <= 10) 
    cout << "YES" << endl;

-check: x=0? (1<=0 && 0<=10) => false && true
-check: x=5? (1<=5 && 5<=10) => true && true
-check: x=100? (1<=100 && 100<=10) => ??
```