cis1.5 spring2009 lecture II.1

today we are going to talk about...

reading simple information and making decisions about it

- writing to the screen ("output") using `cout`
- reading from the keyboard ("input") using `cin`
- relational operators
- branching statements

writing to the screen using `cout`

- `cout`
  - this is called a *method*
  - it is a standard part of the C++ language
  - it produces output on the computer screen
- *arguments*
  - << followed by a *string*, i.e., text in double quotes (")
  - tell `cout` what to write on the screen
- example:

```
#include <iostream>
using namespace std;
int main() {
  cout << "hello world!" << endl;
} // end of main()
```


reading from the keyboard: using `cin`

- `cin`
  - this is also a *method* that is a standard part of the C++ language
  - it reads input from the keyboard
- *arguments*
  - >> followed by a *variable*
  - tell `cin` what to read from the keyboard and where to store it
- example:

```
#include <iostream>
using namespace std;
int main() {
  int i;
  cout << "enter a number: ";
  cin >> i;
  cout << "you entered: " << i << endl;
} // end of main()
```


more on `cin`

- you can use `cin` to read integers and characters, as well as other simple data types (e.g., float, double)
- if you want to read in multiple values, you can use multiple calls to `cin` or one call with multiple parameters
- example of making multiple calls:

```
int i, j, k;
cout << "enter three numbers: ";
cin >> i;
cin >> j;
cin >> k;
```

- when you put this code in a program and run it, you can enter three numbers separated by *whitespace* (space, tab or return)

- example of making one call with multiple parameters:

```
int i, j, k;
cout << "enter three numbers: ";
cin >> i >> j >> k;
```

- as above, when you put this code in a program and run it, you can enter three numbers separated by *whitespace* (space, tab or return)

relational operators

- relational operators are used to compare two values
- they can be used to compare numbers or characters
- comparing characters uses the ASCII table (remember asciimation?)
- the relational operators look like operators in math, except for equality:

==	equality	!=	inequality
>	greater than	<	less than
>=	greater than or equal to	<=	less than or equal to

- examples:

```
x < y
a > b
```

- relational operators are used as part of statements
- one kind of statement is a *branching statement*...

decision making

- *branching statements* are used to allow computer programs to *make decisions*
- `if` a statement is true, then do one thing; otherwise, do something else
- you make decisions like this all the time:

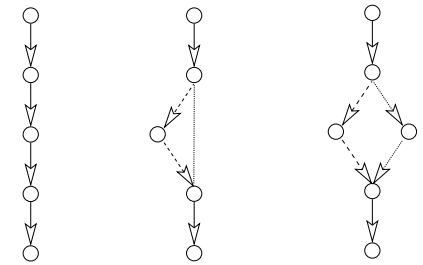
```
If the Q train is in Newkirk station when my B train arrives,
then I will run across the platform and catch the Q train to Avenue H;
otherwise, I will walk to Avenue H
```

- a computer program can make the same types of decisions
- and frequently these are made using relational operators...
- example:

```
if ( x > y ) {
  cout << "x is bigger than y\n";
}
else {
  cout << "y is bigger (or the same as x)\n";
}
```


branching statements

- the `if` statement is part of the C++ language. it is a type of *control structure*, which means that the program control can move from one "branch" to another, instead of always taking a single path.



- there are three forms of the `if` statement in C++: (1) simple `if`, (2) `if-else`, and (3) `if-else-if`

the simple `if` statement

- syntax:

```
if ( <something is true> ) {
  <follow some instructions>
}
```

- example:

```
if ( x > y ) {
  cout << "x is bigger than y\n";
}
```


the `if-else` statement

- syntax:

```
if ( <something is true> ) {
  <follow some instructions>
}
else {
  <follow some other instructions>
}
```

- example:

```
if ( x > y ) {
  cout << "x is bigger than y\n";
}
else {
  cout << "y is bigger (or the same as x)\n";
}
```


the `if-else-if` statement

- syntax:

```
if ( <something is true> ) {
  <follow some instructions>
}
else if {
  <follow some other instructions>
}
else if {
  <follow other, different instructions>
}
else {
  <follow even different instructions>
}
```


- example:

```
if ( x > y ) {
  cout << "x is bigger than y\n";
}
else if ( y > x ) {
  cout << "y is bigger\n";
}
else {
  cout << "y is the same as x\n";
}
```