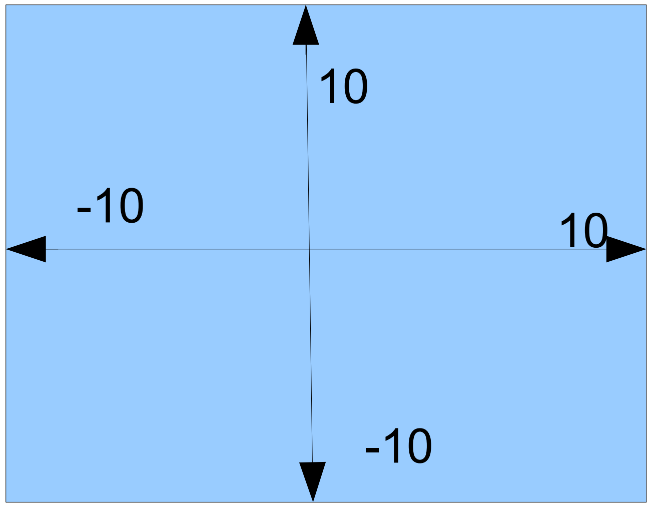
PIC 10A

Lecture 7: Graphics II and intro to the if statement

Setting up a coordinate system

By default the viewing window has a coordinate system already set up for you



The origin is in the middle of the screen

$-10 < x < 10$

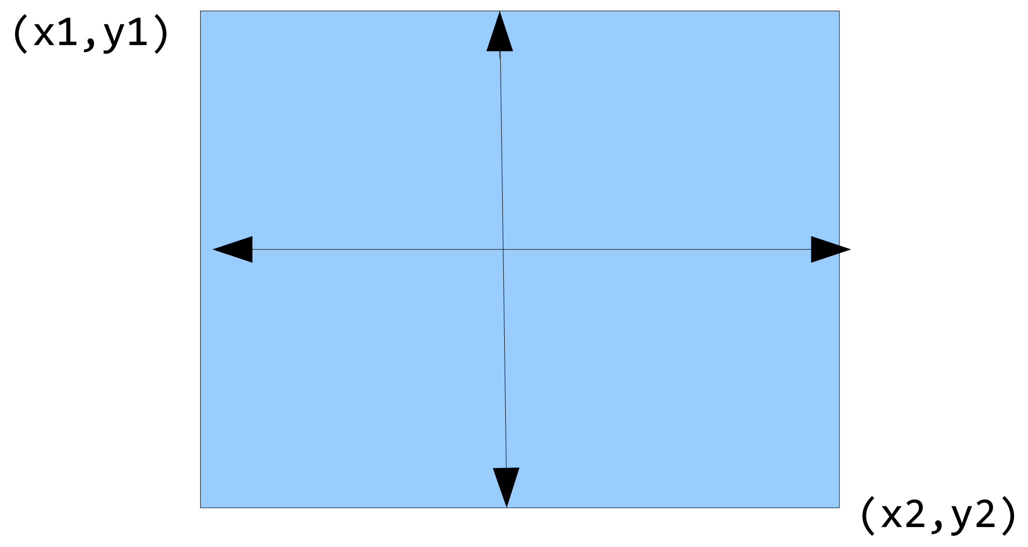
$-10 < y < 10$

Setting up coordinates

To set up your own coordinates you can write:

```
cwin.coord(x1,y1,x2,y2);
```

Where x1 and y1 are coordinates of the top left point and x2 and y2 are coordinates of bottom right point.



```
cwin.coord(-20,20,20,-20);
```

The Input Window & Prompts

All input and prompts are displayed at the top of the graphics window.

The prompt appears in a gray box.

Plan your graphics to leave some blank space at the top, or else they'll be covered up when a prompt appears.

Like the Message class, the prompt could be any string, but escape characters (like \n \t) won't work.

The Input Window & Prompts

To get text input from the user we use cwin's member function get_string.

Example:
```
string text;
text = cwin.get_string("Type some text");
```

get_string does three things:

1) It displays the text in quotes on the graphics window.
2) It waits for user to enter text
3) It grabs the text and returns it.

Because get_string does not automatically put the input text into some variable like cin does, it just returns it, so we need to assign the returned value to some variable.

Since ccc_win uses the <string> library, we don’t need to include it to use strings.

The prompt and user input appears at the top of the graphics window. We can’t move it.
If you want text inside the window, use Message
class.

Example

```
#include "ccc_win.h"
int ccc_win_main ( )
{
    string first_name;
    string last_name;
    string message_string;

    first_name = cwin.get_string ("What is your first name?");
    last_name = cwin.get_string ("What is your last name?");

    message_string = "Your full name is: " + first_name + " " +
last_name;

    cwin << Message (Point(0,1), message_string);


    return 0;
}
```

cwin cont.

Similarly we can get numbers from the user:

int

```
int age = cwin.get_int("How old are you?");
```

double

```
double x_coord = cwin.get_double("Enter the x-coordinate.");
```

Getting mouse click location

We can also get as an input a location where the user clicks.

```
Point P = cwin.get_mouse("Click somewhere!");
```

- `get_mouse` returns the location as a Point object.
- The prompt will appear at the top of the screen.
- The coordinates are displayed on top of the screen as mouse moves.

Example: User created triangle

```
#include "ccc_win.h"
int ccc_win_main ( )
{

    Point P1;
    Point P2;
    Point P3;

    P1 = cwin.get_mouse("Click point 1!");
    P2 = cwin.get_mouse("Click point 2!");
    P3 = cwin.get_mouse("Click point 3!");

    Line L1(P1,P2);
    Line L2(P2,P3);
    Line L3(P3,P1);

    cwin << L1 << L2 << L3;

    return 0;
}
```

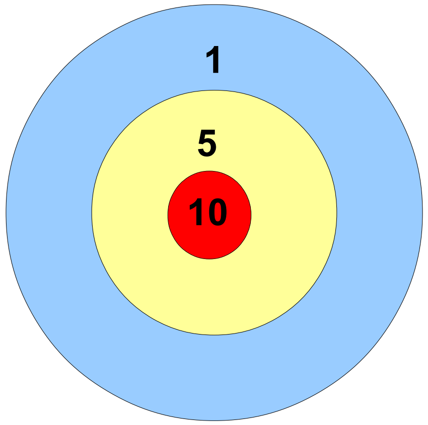
cwin member functions

`cwin.coord (double x1, double y1, double x2, double y2)`	Sets the viewing window with top left corner (x1,y1) and bottom right corner (x2,y2).
`cwin << x`	Outputs graphics object x (Point, Line, Circle, or Message).
`cwin.clear ( )`	Erases the screen.
`string cwin.get_string (string text)`	Displays prompt text and returns the entered string.
`int cwin.get_int (string text)`	Displays prompt text and returns the entered integer.
`double cwin.get_double (string text)`	Displays prompt text and returns the entered double.
`Point cwin.get_mouse (string text)`	Displays prompt text and returns the mouse click point.

Hitting the target

Consider a following hypothetical program.

We want to draw concentric circles and then label each circle with a number as shown in the figure below.



Then we want to prompt the user to click on the target and depending on which circle they clicked, we want to tell them what their score is.

Hit the target pseudo code

1. Make three concentric circles using the circle class.
2. Draw the circles.
3. Create three messages for the score labels.
4. Write the messages to the screen.
5. Prompt the user to click on the screen and capture the point where they click into a point variable.
6. Figure out the users score.
7. Create a message with the users score.
8. Write the score message to the window.

If statement

How do we figure out how many points to report?

It depends on which circle we are inside.

We want our code to function something like the following:

```
score = 0
if (inside biggest circle)
    score += 1
if (inside middle circle)
    score += 4
if (inside smallest circle)
    score += 5
```

How does an if statement work?

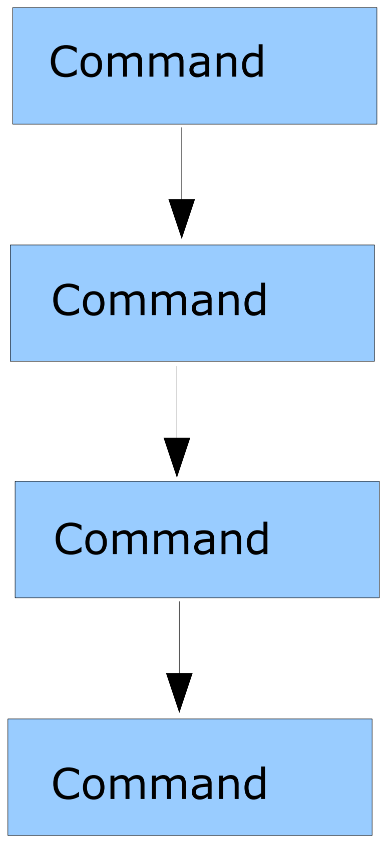
```
if (expression)
{
  // Execute the code between braces
}
```

If the expression is true the code between the braces is executed.

If the expression is false, all the code in the braces is skipped.

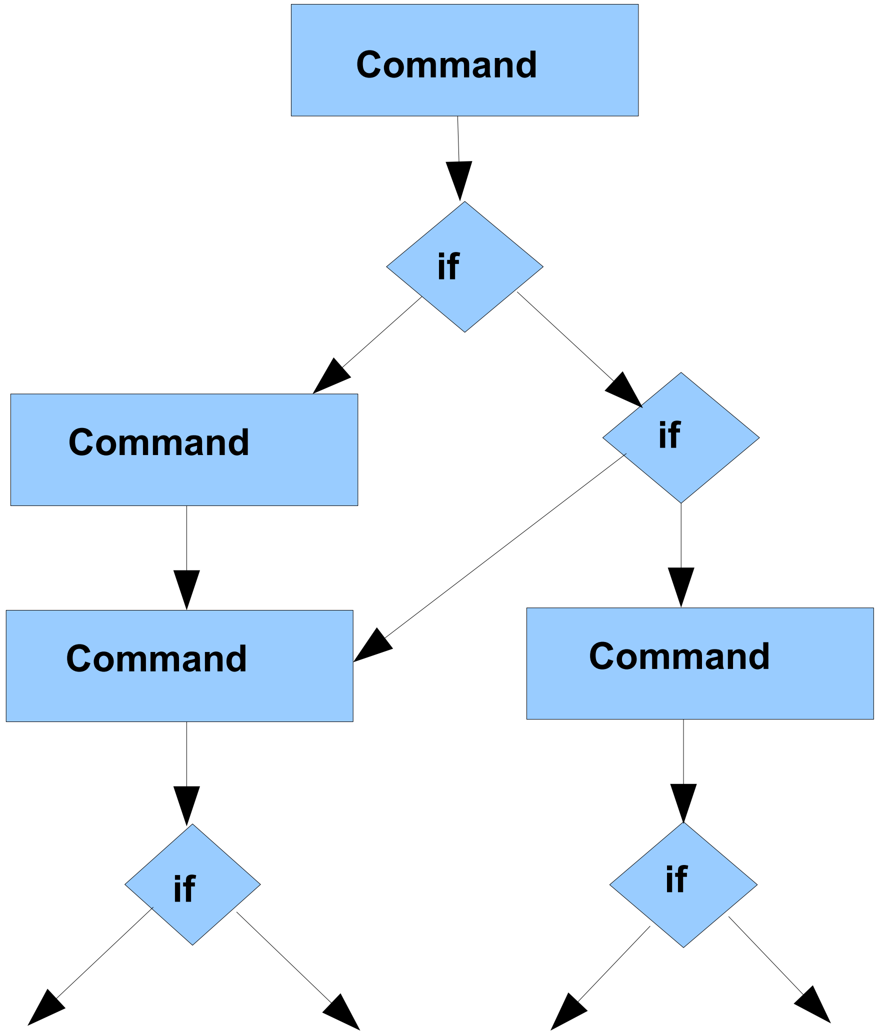
Why are if statements useful?

Here is a flow diagram for all our programs before if statement



Pretty boring...

Flow chart for a typical program with if statements



Boolean expressions

Boolean expression is something that the computer can assign a true or false value to.

Example:

6 > 2

3 > 17

Boolean variable is a variable that can only store one of two values: true (1) or false (0).

`if` statements use boolean expressions!

```
if (boolean expression)
{
    //Do stuff
}
```

Table of basic boolean operators

Operator	Description	True example	False example
<	Less than	2 < 5	5 < 2
>	Greater than	5 > 2	2 > 5
>=	Greater than or equal	5 >= 5	2 >= 5
<=	Less than or equal	2 <= 5	5 <= 2
==	Equal (comparison)	2 == 2	2 == 5
!=	Not equal	2 !=5	2 !=2

`if` statement

When the computer encounters an if statement it checks first if the boolean expression in the parenthesis is true. If it is true then the computer executes the code inside the braces.

```
if (boolean expression)
{
  //Do stuff
}
```

Example

```
int x;

cout << " Please give me a number";
cin >> x;

if (x > 5)
{
cout << "Your number was greater than 5";
}
```

Importance of { }

Actually ommitting the braces is legal.

We could write:

```
if ( x > 7 )
  cout << "Your number is bigger than 7!";
```

The program works as expected.

If you omit braces the next line of code is assumed to be under the if statement.

Importance of { }

what about:

```
if ( x > 7)
    cout << "Your number is bigger than 7!";
    cout << "Congratulations you win!";
```

Here regardless of value of x the computer will output the line:
Congratulations you win!

When you want multiple statements under an if statement, you need braces.

```
if ( x > 7)
{
    cout << "Your number is bigger than 7!";
    cout << "Congratulations you win!";
}
```

Good idea to always use the braces because it doesn't hurt.