Today in CS161

- ***Week #7 Arrays***
  - **Learning about arrays**
    - Being able to store more than one item using a variable
  - **Examples**
    - Tic Tac Toe board as an array
  - **Graphical**
    - User interaction for the tic tac toe program

Why we need….Arrays

- Think for a moment what it would be like if you wanted to store your name or all of the scores for your homework?
- My first name is 5 characters long – I really don't want 5 character variables. Or, what about my middle name! Way too long.
- Or, think about homework scores: 6 homework plus a midterm and final to consider. Does that mean we would need 8 variables if we wanted to write a program to keep track of our grade in the class? **No!**
- Our programs would become large and hard to manage.

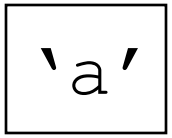
First…Arrays of Characters

- For a name, we have a choice of using..
  - A string
  - Or, an Array of characters
- In C, you have to use an array of characters for a string
- In Java, you typically use a string
- In C++ we find that not everything can be done with a string, so we will learn about arrays first…then strings.
- We all know what a character is (a single byte), so what's an array of characters?
  - a sequence of character stored sequentially in memory

How do I define an Array of Characters?

- We know how to define a single character:

```
char ch='a';
```



- But what about an array of characters?

```
char str[5];
```

- Since these are just characters stored sequentially in memory, we use a special character to indicate the end of a string: `'\0'`

How do I read in a string?

- There are two ways to read in strings
- If the string is a sequence of characters without any whitespace (like your first name), then you can say:

```
cin >> str;
```

- If I enter “hi”, this is what is stored:

What does `cin >> array_of_characters` do?

```
char str[5];
cin >> str;
```

- When reading in an array of characters, cin and the extraction operator (>>) skip leading whitespace and read characters until a whitespace character is encountered.
- Then, it automatically stores a '\ 0' after the last character read in.

What do we need to be careful about?

- We need to be careful when working with arrays of characters...
- If we have an array of size 5
  - that means there are 5 bytes of memory allocated for our variable sequentially in memory
- This means that we can store four characters at most, since one spot needs to be reserved for the **terminating nul**

So, What could happen???

- Using cin >> str;
- If I enter “hello”, what is stored?

- Notice we ended up storing the ‘\0’ in memory that is not allocated for our variable
  - this is extremely dangerous and can cause our programs to bomb! (segmentation fault or core dump when running...)
  - It could stomp on memory of another variable!!

What do we need to be careful about?

- What this means is that C++ does not check to make sure we stay within the bounds of our arrays
- C++ assumes that we know what we are doing!
- It is a powerful language...one that can even be used to design operating systems
- Therefore, if there is a chance that the user may type in too many characters, we need to read in our strings using a different approach

Side note…what does cin.get() do?

- There is a cin.get function that is useful
- There are three ways to use this function:
  - it can be used to read a single character

    char ch; ch = cin.get();

    cin.get(ch);

    this reads in the next character from the input buffer, even if that next character is whitespace!

How do I read in an array of characters safely?

- Or, we can use the cin.get function to read in an array of characters using 2 or 3 arguments:

```
char str[5];
cin.get(str, 5);
cin.get(str, 5, '\n');
```

this reads in the next sequence of characters up until (size-1) characters are read the delimiting character is encountered n' by default)…but not read

The 3 argument cin.get function

- The three argument version of cin.get has the following form:

```
cin.get(array_name, max_size,
        delimiting_character);
```

- A side benefit of this function is that it will allow us to read in sentences, our entire first/last name, a paragraph, etc. This is because the delimiting character need not be white space!

Reading in an entire line…or name…

- There is one “gotcha” with this function.
- While the three argument version of cin.get won’t read in too many characters (so it will never store characters outside your array bounds),
  - it will not read in the delimiting character!
- Therefore, we must always “eat” the delimiting character, using either:

```
cin.ignore();     or     cin.ignore(100,'\n');
cin.get();        or     while(cin.get()  !=  '\n');
```

Ignoring the delimiter…

cin.ignore(); *Ignores the next character*

cin.ignore(100,'\n');

Ignores the next 100 characters or until the newline is read in

cin.get(); *Reads (and throws out) the next character…since we are not saving it in a variable*

Ignoring the delimiter…

```
while(cin.get()  !=  '\n');
```

This reads a character. Checks to see if it is a newline. If it isn't, the body of the loop is executed (which does nothing). Then it reads again…it continues the process until a newline is read

Ignoring the delimiter…

```
while(cin.get()  !=  '\n');
```

Is the same as….

```
char ch;
do
{
   ch = cin.get();
} while (ch != '\n');
```

Example using cin.get:

- Using `cin.get(str, 5);`
- If I enter "hi !", what is stored?

- Notice that room is left to store the '\ 0' at the end of the array, and there is no danger of writing outside of our array bounds.
- But, what is left in the input buffer? `'\n'`
- How do we "flush" this? **cin.ignore();**

Another example using cin.get:

- Using `cin.get(str, 5);`
- If I enter "hello what is stored?

- Notice that room is left to store the '\ 0' at the end of the array, and there is no danger of writing outside of our array bounds.
- But, what is left in the input buffer? `'o\n'`
- How do we "flush" this? **while(cin.get() !='\n');**
  - **Or cin.ignore(100,'\n');**

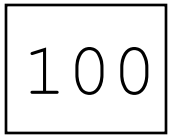
How do I display an array of characters?

- Luckily, displaying strings isn't as complicated. `cout << str;`
- Simply by using cout followed by the insertion operator (<<), we can display as many characters as have been stored in the array until the terminating nul ('\0') in encountered.
- Notice, the '\0' is important so that we don't display "garbage" characters (i.e., memory that has not been set or used yet!)

What about other kinds of arrays?

- **We can create an array of any kind of data you would like**: *one int:*

  int value = 100



- But what about an array of integers?

```
int scores[5];
```

- These are

  just integers stored sequentially in memory.

*** THERE IS NO SPECIAL ENDING Character for arrays of any type other than arrays of characters.

Working with arrays – element by element

- Unlike arrays of characters, any other type of array must be worked with one element at a time:
  - by indexing through the array
  - we begin by using subscripts that start at zero and then progress until the array size-1
- For example, we can read in an array of integers one element at a time:

```
for (int i = 0; i < 5; ++i)
{
        cout <<“Enter another score: “;
        cin >> score[i];
}
```

Outputting the Array Contents.

- And, just like inputting … we must output each element of the array using
  - Subscripts
  - score[0] is the first score
  - Score[1] is the second score
  - Score[4] is the last of the five scores
- For example, we can output them….

```
for (int i = 0; i < 5; ++i)
{
      cout <<score[i] <<endl;
}
```

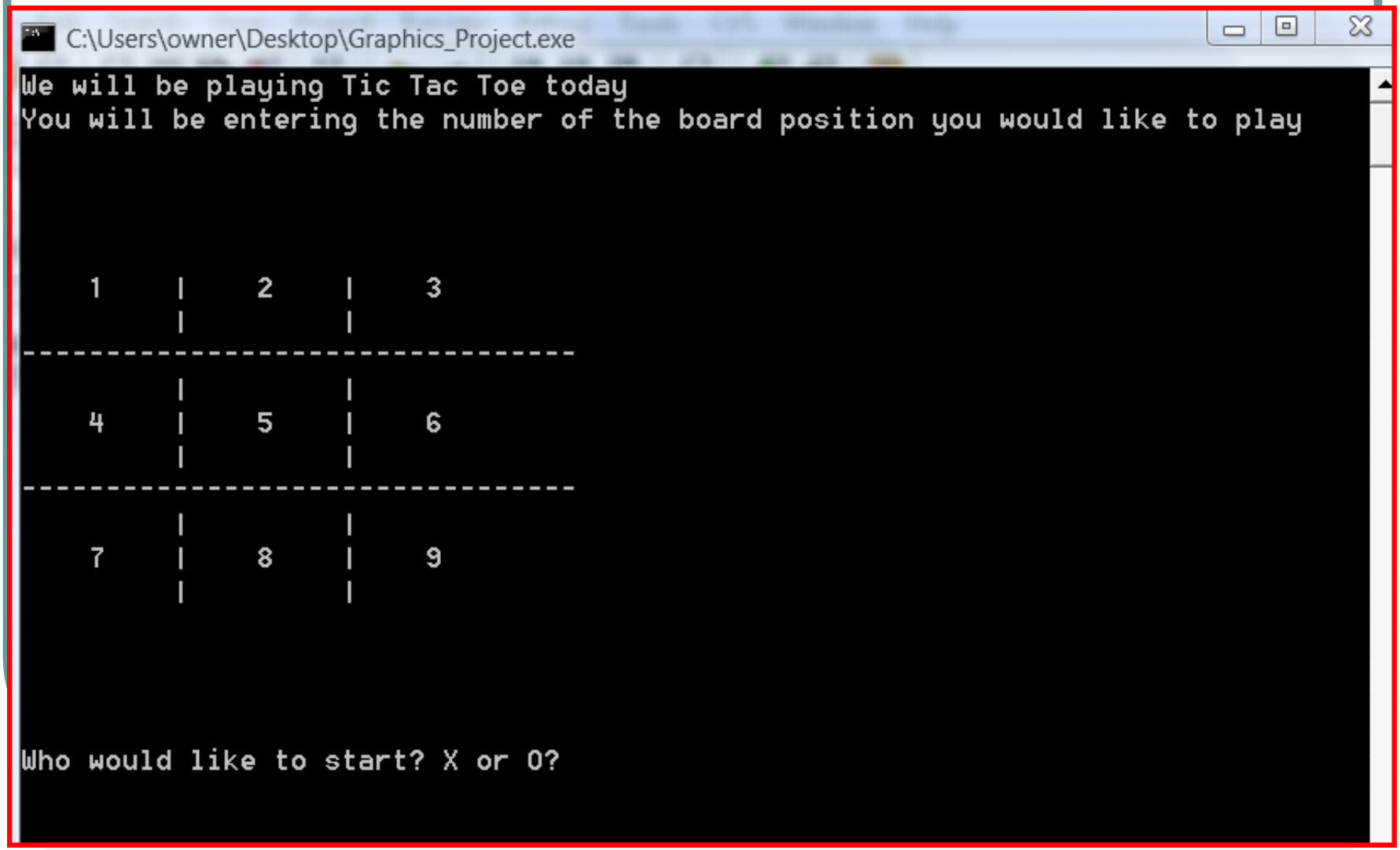
Examples…Non Graphical

- ***Tic Tac Toe***
  - ***Using arrays***
  - ***But first, we won't draw anything***
  - ***Output vertical bars on your non-graphical screen for the board***

Tic Tac Toe

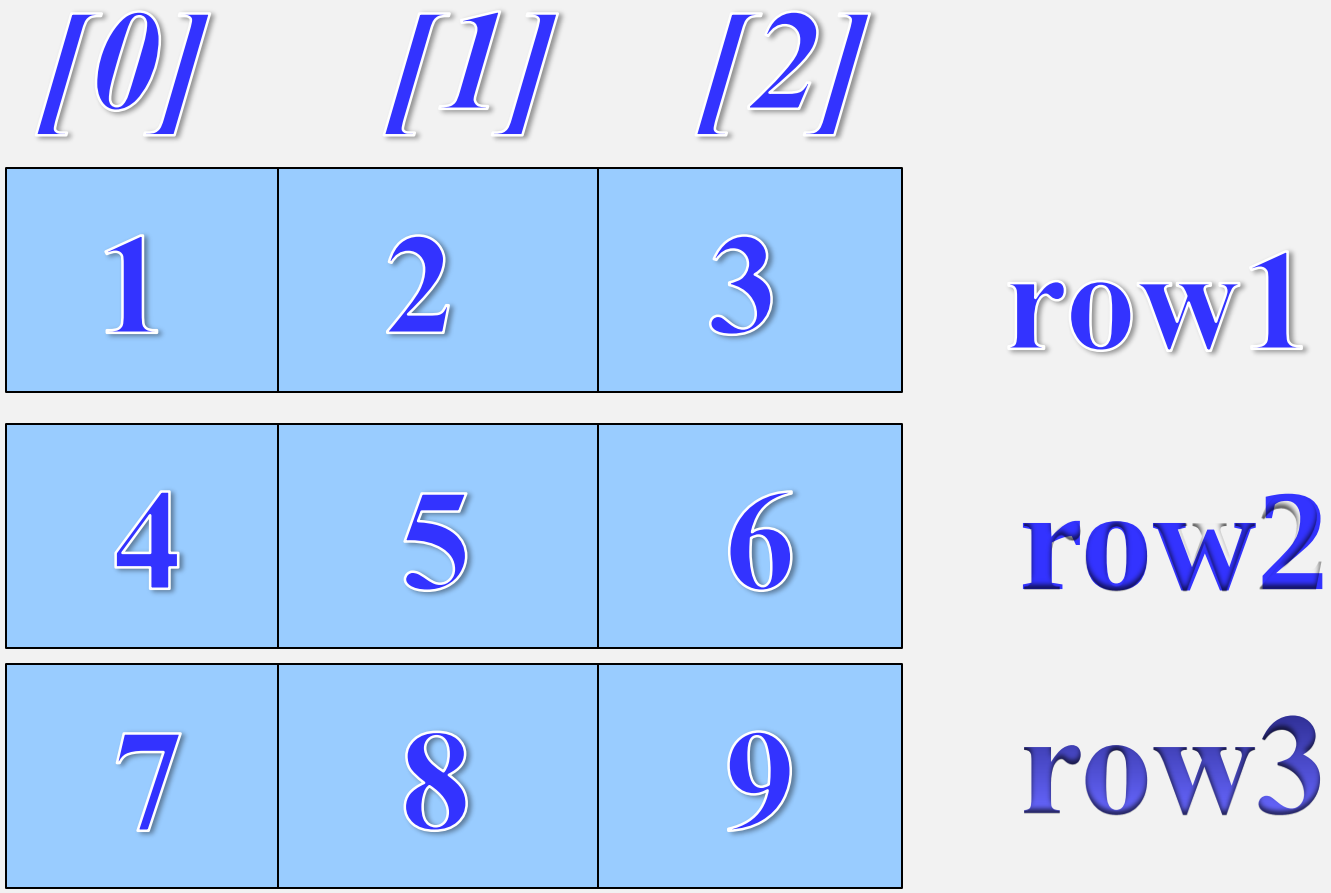
- ***Practices using Functions***
- ***Keeps main small***
- ***And, uses 3 arrays to keep track of the player's X, O choices***
- ***Algorithm:***
  - ***Display board, select first player***
  - ***Get the player's choice until correct***
  - ***Show their choice on the board***
  - ***Switch users, unless there is a winner or cat scratch***
  - ***Continue until we are done (winner or cat scratch)***
  - ***Display the appropriate message***

Tic Tac Toe – without graphics!



How are the arrays used?

```
char row1[4]={'1','2','3'};     //create the initial positions available
char row2[4]={'4','5','6'};
char row3[4]={'7','8','9'};
```



Displaying the board

```
//Draw the tic tac toe board
void drawboard()                                    //uses the #include <iomanip>  library
{
    cout <<endl <<endl;
    cout <<setw(5) <<row1[0] <<setw(5) <<'|' <<setw(5) <<row1[1] <<setw(5)
              <<'|' <<setw(5) <<row1[2] <<endl;
    cout <<setw(10) <<'|' <<setw(10) <<'|' <<endl;
    cout <<"-------------------------------" <<endl;
    cout <<setw(10) <<'|' <<setw(10) <<'|' <<endl;
    cout <<setw(5) <<row2[0] <<setw(5) <<'|' <<setw(5) <<row2[1] <<setw(5)
              <<'|' <<setw(5) <<row2[2] <<endl;
    cout <<setw(10) <<'|' <<setw(10) <<'|' <<endl;
    cout <<"-------------------------------" <<endl;
    cout <<setw(10) <<'|' <<setw(10) <<'|' <<endl;
    cout <<setw(5) <<row3[0] <<setw(5) <<'|' <<setw(5) <<row3[1] <<setw(5)
              <<'|' <<setw(5) <<row3[2] <<endl;
    cout <<setw(10) <<'|' <<setw(10) <<'|' <<endl;
}
```

Easier….

```
//Draw the tic tac toe board
void drawboard()
{
    cout <<endl <<endl;
    cout <<"    " <<row1[0] <<"    " <<'|' <<"    " <<row1[1] <<"    " <<'|' <<"    " <<row1[2]
      <<endl;
    cout <<"        " <<'|' <<"        " <<'|' <<endl;
    cout <<"----------------------------" <<endl;
    cout <<"        " <<'|' <<"        " <<'|' <<endl;
    cout <<"    " <<row2[0] <<"    " <<'|' <<"    " <<row2[1] <<"    " <<'|' <<"    " <<row2[2]
      <<endl;
    cout <<"        " <<'|' <<"        " <<'|' <<endl;
    cout <<"----------------------------" <<endl;
    cout <<"        " <<'|' <<"        " <<'|' <<endl;
    cout <<"    " <<row3[0] <<"    " <<'|' <<"    " <<row3[1] <<"    " <<'|' <<"    " <<row3[2]
      <<endl;
    cout <<"        " <<'|' <<"        " <<'|' <<endl;
}
```

Tic Tac Toe – Selecting a player

- ***Selecting the first player***

```
//Who is the first player?
char firstplayer()
{
    char xo;
    cout <<endl <<endl <<"Who would like to start? X or O? ";
    cin >> xo;   cin.get();
    xo = toupper(xo);      //let's keep it upper case only!
    while (xo != 'X' && xo != 'O')
    {
        cout <<"Sorry - you need to enter an X or an O. Try again! ";
        cin >>xo;    cin.get();
        xo = toupper(xo);
    }
    return xo;
}
```

Time to select a board location

```
//Select a location on the board
int selectloc(char player)  //the argument is an X or an O
{
   int selection;
   drawboard();   //show the locations on the board to select from
   cout <<endl <<endl <<"Player: " <<player;
   cout <<"...please enter your selected board location: ";
   cin >> selection;    cin.get();
   while (selection < 1 || selection > 9)
   {
        cout <<"Sorry - enter a number between 1 and 9. Try again! ";
        cin >>selection;   cin.get();
   }
   return selection;    //returns a valid number!
}
```

Look at the individual row….

- ***Since the same operations happen on each row…I wrote a function***
  - ***And…passed the appropriate row to be looking at as an argument***
  - ***Remember position is between 1-9…but array indices for our three row arrays are 1,2,3***

```
bool checkloc(int position)
{
    if (position >=1 && position <= 3)      //row 1
       return checkrow(row1,position-1);
    if (position >= 4 && position <= 6)     //row 2
        return checkrow(row2,position-4);
    return checkrow(row3,position-7);       //row 3
}
```

Check the location first….

- ***Is the requested location being used?***

```
//Check to see if the position on the board has been taken?
//We need to check the appropriate row

bool checkrow(char row[], int index)
{
    if (row[index] == 'X' || row[index] == 'O')
    {
       cout <<"That spot has been taken already - Try again!";
          return false;
    }
    return true;
}
```

Store the request “on” the board

- ***If the position was valid***
  - ***And if the position request was not already being used***

```
//At this point we know we have a valid position so just play it!
void playposition(int position, int player)
{
    if (position >=1 && position <= 3)      //row 1
      row1[position-1] = player;
    else if (position >= 4 && position <= 6) //row 2
      row2[position-4] = player;
    else
       row3[position-7] = player;           //row 3
}
```

Continue to play until…

- ***There is a winner!***

```
//Find out if there is a winner - horizontal, vertical, or diagonal
bool winner()
{
    //Is there a horizontal winner?
    if (all_across(row1) || all_across(row2) || all_across(row3))
      return true;
    //Is there a vertical winner?
    if (all_down(0) || all_down(1) || all_down(2))
      return true;
    //Is there a diagonal winner?
    if (all_diag())
      return true;

    return false;
}
```

Check across for a row…

- ***Functions allowed us to reuse the same code for each row:***

```
//Are the same values all across a row?
bool all_across(char row[])
{
    if (row[0] == row[1] && row[0] == row[2]) //all the same
       return true;
    return false;
}
```

Check down for a column…

- ***Functions allowed us to reuse the same code for each col:***

```
//Are the same values in a columm?
bool all_down (int col)
{
    if (row1[col] == row2[col] && row1[col] == row3[col])
       return true;
    return false;
}
```

Check both diagonals…

- ***Functions allowed us to keep the winner function simple:***

```
//Are the same values diagonal?
bool all_diag()
{
    if (row1[0] == row2[1] && row1[0] == row3[2])
      return true;
    if (row1[2] == row2[1] && row1[2] == row3[0])
      return true;
    return false;
}
```

We could also be done if…

- ***There was a cat scratch!***

```
//Is there a catscratch?
bool catscratch()
{
    //Are all of the places filled up?
    if (no_holes(row1) && no_holes(row2) && no_holes(row3))
       return true;
    return false;  //we can still play
}
```

Again, functions allowed me to reuse the same code for row1 then for row2 and again for row3

Finding if there are holes or not...

- ***There was a cat scratch!***

```
//Find out if there are any holes in the current row
bool no_holes(char row[])
{
    for (int i = 0; i < 3; ++i)
      if (row[i] != 'X' && row[i] != 'O')
        return false; //there are still places to play!
    return true;
}
```

Other functions…

- ***Switching the player***

```
//Switch from an X to an O
void switchplayer(char & player)
{
    if (player == 'X')
       player = 'O';
    else
        player = 'X';
}
```

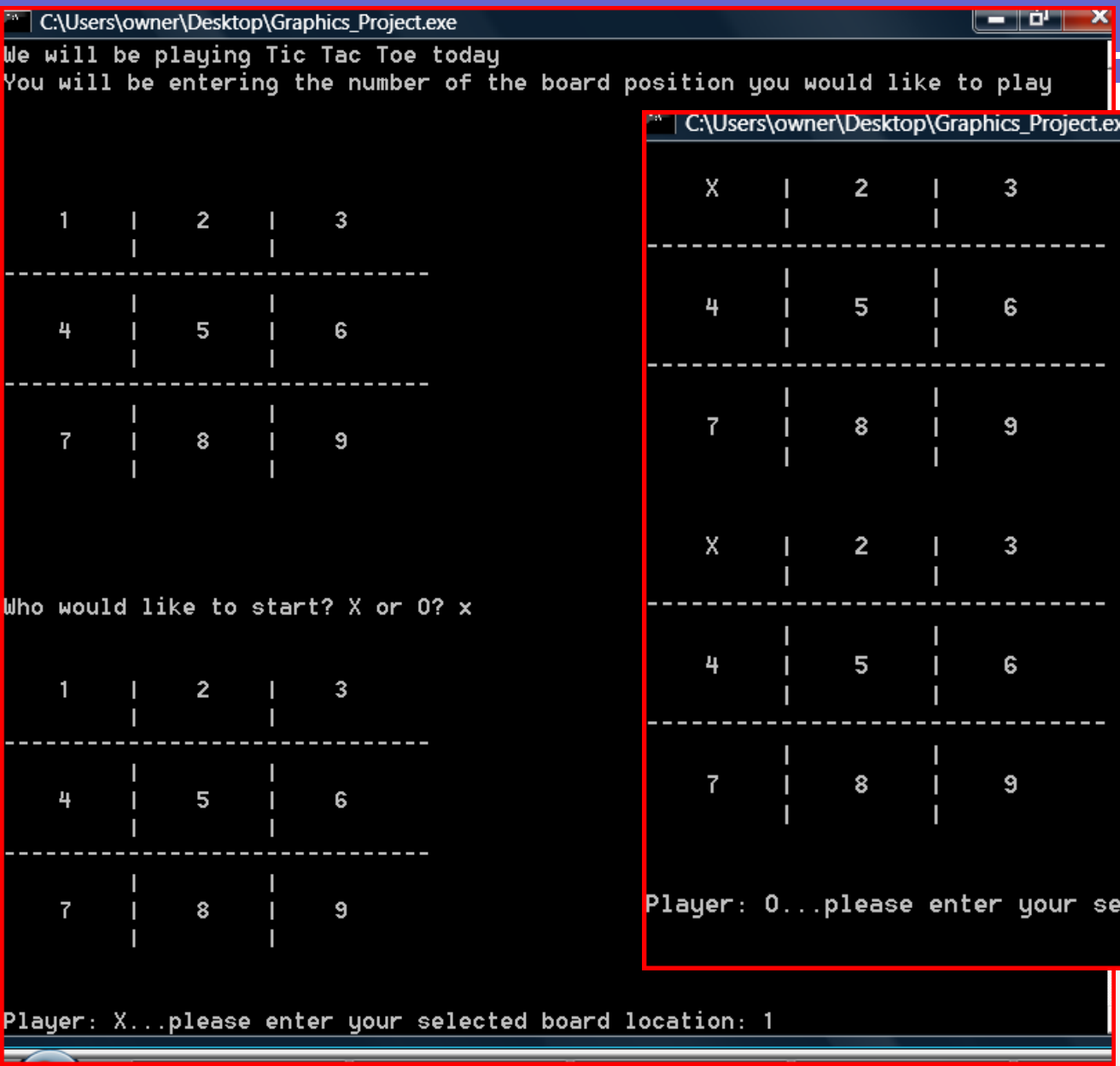
What main could look like…

```
int main()
{
   char player;            //X or O
   int pos;                //1-9 position

   welcome();              //tell the user the rules
   player = firstplayer();  //who goes first?

   do      //start playing until there is a winner
   {
         do
         {
            pos = selectloc(player); //Where to play?
         } while (!checkloc(pos));   //Is it a valid choice?
```

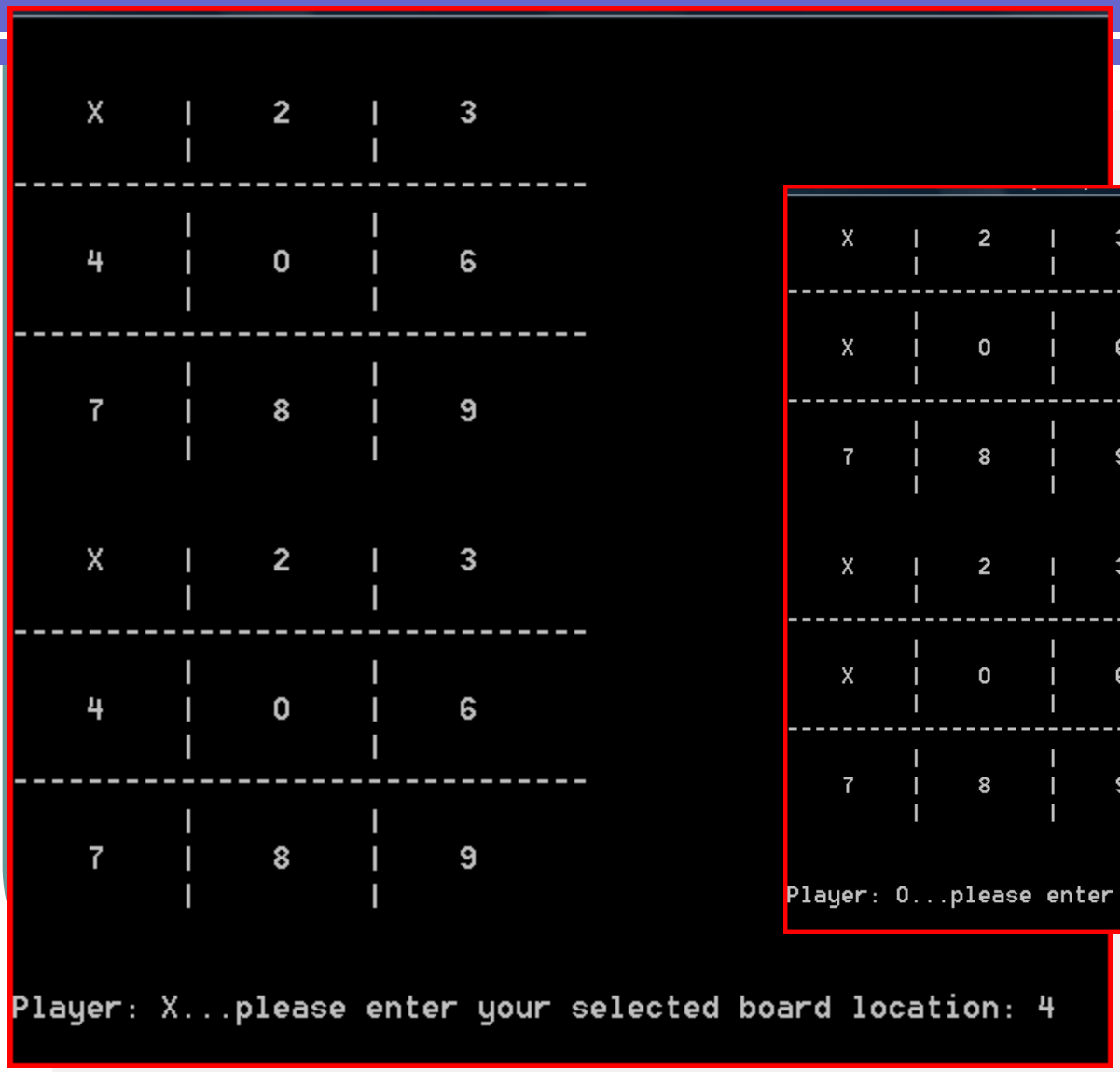
What main could look like…

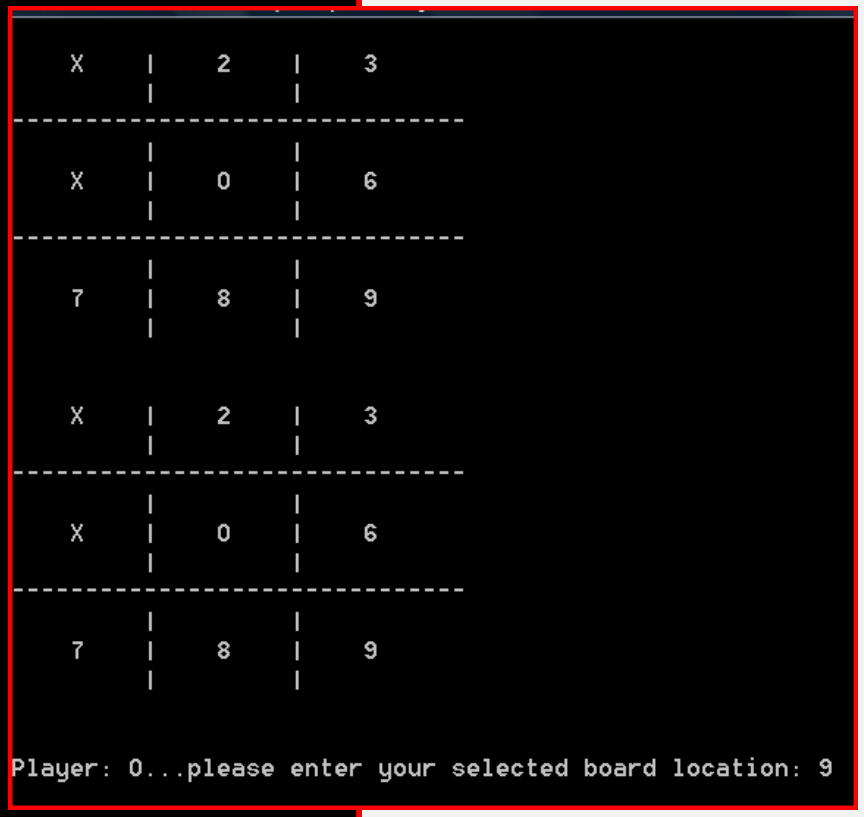
```
//OK, now we have a valid choice, let's put it on the board
            playposition(pos, player);
            drawboard();
            if (!winner() && !catscratch())  //don't switch if we are done
               switchplayer(player);    //Next person's turn!

    } while (!winner() && !catscratch());  //are we done yet?

    if (winner())
       cout <<"CONGRADULATIONS! Player " <<player <<" WINS!!!!!!!!!! "
                  <<endl;
    else if (catscratch())
        cout <<"CAT SCRATCH...Ouch! Play again sometime!" <<endl;
    cin.get();
    return 0;
}
```

Playing the game…

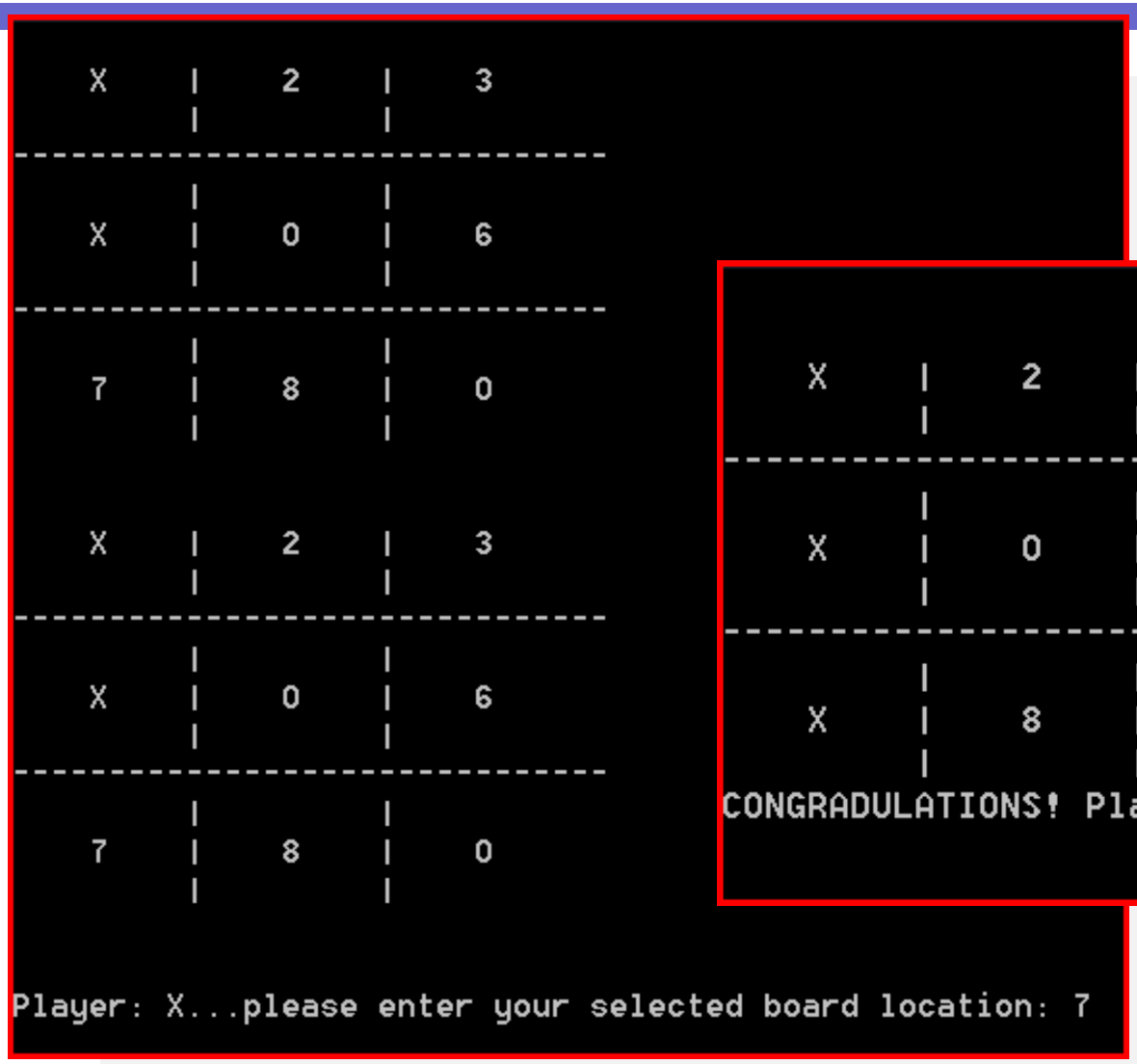


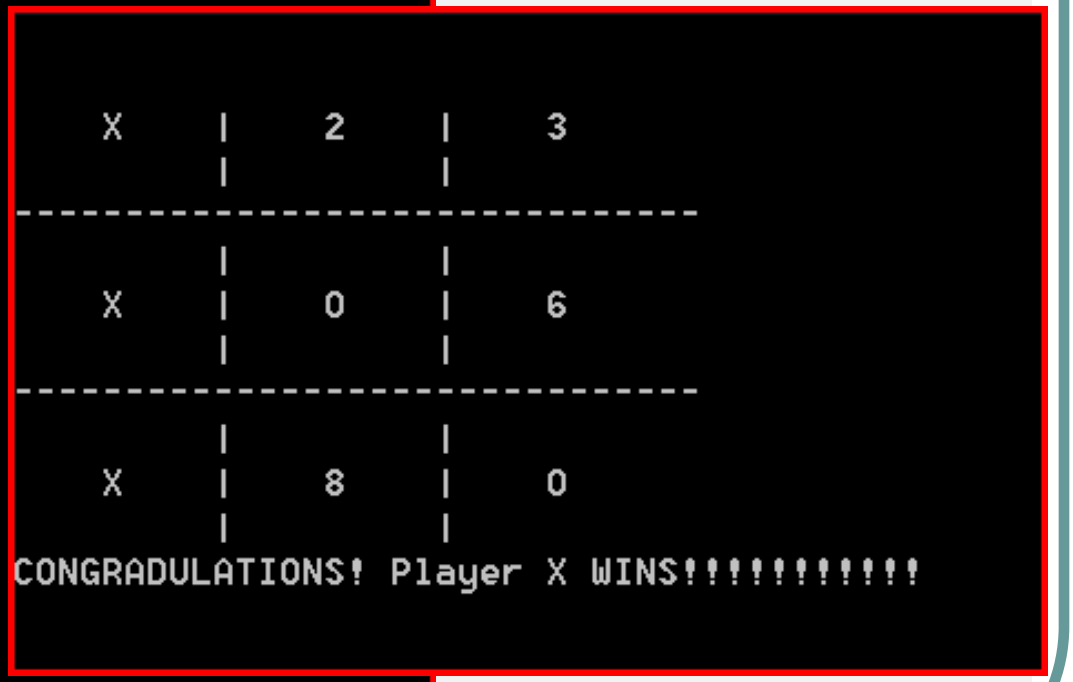
Playing the game…





Playing the game…





Today in CS161

- ***Next Topic: Arrays of Characters***
  - **Using Arrays of Characters**
    - Copying
    - Comparing
  - **Examples**
    - Working with arrays of characters using small examples
  - **Graphical**
    - User interaction

Operations on Arrays of Characters

- There are very few operations that can be performed on array of characters (i.e., strings)
- We can read in string using:

```
cin >> array_of_characters;
cin.get(array, size, delimiter);
```

- We display strings using:

```
cout << array_of_characters;
```

- But, there are no others...

Operations on Arrays of Characters

- For example, we can't compare two arrays of characters by saying:

```
char str1[10], str2[10];
if (str1 == str2)
```

- This is because an array is really *the address of the first element in a sequentially allocated set of memory*.
- So, the == or != operators would simply be comparing the memory addresses! Oops!

Comparing Arrays of Characters:

- Instead, to compare two arrays of characters we must include another library: cstring
- And, call the string compare function:

```
strcmp(first_array, second_array);
```

- The strcmp function returns:

  0 if first_array is equal to second_array

  <0 if first_array is less than second_array

  >0 if first_array is greater than second_array

Copying Arrays of Characters

- We also copy strings using the assignment operator:

```
char str1[10], str2[10];
str1 = str2;
```

- This is illegal because an array is really *the constant address of the first element of the array*. *We can't change the location in memory where your array is located!!!! And…that is what this assignment statement is attempting to do…*
- Instead, we call strcpy from cstring library:

```
strcpy(str1, str2);//str1=str2;
```

For example:

- Let's now put this to use by writing a function to read in two strings and displaying them in alphabetical order
- First, write the algorithm:
  - Get two strings (prompt, input, echo)
  - If the first string is less than the second
    - display the first string followed by the second
  - If the first string is greater or equal to the second
    - display the second string followed by the first

Sorting Two Names:

```
#include <cstring>
void sort_two() {
   char first[20], second[20];
      cout << "Please enter two words, one per
   line: ";
   cin.get(first,20, ' ');
   cin.get();       //don't forget this part!
   cin.get(second,20, '\n');
   cin.get();       //eat the carriage return;
   if (strcmp(first, second) < 0)
       cout << first << ' ' << second << endl;
   else
       cout << second << ' ' << first << endl;
}
```

Change the function to have args:

```
void sort_two(char first[], char second[]) {
   cout << "Please enter two words: ";
   cin.get(first,20, '\n');    cin.get();
   cin.get(second,20, '\n');
   cin.get();       //eat the carriage return;
   if (strcmp(first, second) > 0) {
       char temp[20];
       strcpy(temp, first);
       strcpy(first, second);
       strcpy(second, temp);
   }
}
```

We'd call the function by saying:

```
void sort_two(char first[], char second[]);
   //prototype

int main() {
   char str1[20], str2[20];

   sort_two(str1, str2);
   cout << str1 << ' ' << str2 << endl;

   //what would happen if we then said:
   sort_two(str2, str1);
   cout << str1 << ' ' << str2 << endl;
   return 0;
}
```

Example: Writing a program

- Let's write a program that reads in a paragraph, one word at a time, and counts the number of times the word "the" is used (it could be capitalized and still count!)
- Algorithm:
  - Welcome the user
  - Read in a word
  - Check to see if the word is "the" or "The"
    - If it is, **increment** a counter
  - While there are more words continue Reading
  - Display the results

Welcome Function

```
//Tell the user the rules of the program
void welcome()
{
    cout <<"Welcome to the Counting Program! " <<endl;
    cout <<"This program will count the number of times you type in ";
    cout <<"the word The/the " <<endl;
    cout <<"When you are done, type in a '@' symbol" <<endl <<endl;
    cout <<"Let's get started...start typing in a pargraph!" <<endl <<endl;
}
```

Read a word Function

```
//read in a word
void read_word(char word[])
{
    cin.width(MAX); //let's make sure to secure memory

    cin >>word;              //skips leading whitespace,
                              //reads in characters
                             //until whitespace is encountered
                                                  //but not read
    cin.get();     //ignore the whitespace
}
```

Check to see if it is a “the” or “The”

```
//check to see if the word is "the" or "The"
int check_word(char word[])
{
    if (strcmp(word,"the") == 0 ||
        strcmp(word,"The") == 0)
       return 1;

    return 0;     //the “else” situation
}
```

Is it Time to End?

```
//Is it time to end (is there a @ character as the next character?
bool time_to_end()
{
    if (cin.peek() == ' ' || cin.peek() == '\n') //are we at whitespace?
       cin.ignore();              //ignore the whitespace...
    if (cin.peek() == '@')    //time to end
    {
       cin.ignore(100,'\n');   //ignore all the rest!
       return true;
    }
    return false;
}
```

Main….Keeping it small!

```
//This is where the program begins
int main()
{
    char aword[MAX];        //holds the current word
    int total = 0;          //holds the running total, starting at zero

    welcome();
    do
    {       read_word(aword);   //get a word from the input buffer
            total += check_word(aword);  //add a 0 or 1 to the total
    } while(!time_to_end());              //is it time to end?

    cout <<"There were " <<total <<" uses of the word 'the' " <<endl;
    cin.get();
    return 0;
}
```

Extending the program: UPPER CASE!

- Now let's modify the program to allow all versions of the word "the" to be used (the, The, THE, THe, etc.)
- We will do this by capitalizing all characters and then just comparing the array with THE

```
void uppercase(char word[])
{
    int length = strlen(word);
    for (int i = 0; i < length; ++i)
        word[i] = toupper(word[i]);
}
```

New version of check_word….

```
//check to see if the word is "the" or "The"
int check_word(char word[])
{
    uppercase(word);
    if (strcmp(word,"THE") == 0)
       return 1;
    return 0;
}
```

Extending the program: Any word!

- Now let's modify the program to count other words…not just "the". Letting it be the user's choice what word to look for!
- Algorithm:
  - Welcome the user…**explain the new rules**
  - **Read in the word to compare**
  - Start by reading in a word of the paragraph
  - Check to see if the **word matches**
    - If it is, **increment** a counter
  - While there are more words continue Reading
  - Display the results

Welcome Function

```
//Tell the user the rules of the program
void welcome()
{
    cout <<"Welcome to the Counting Program! " <<endl;
    cout <<"This program will count the number of times you type in ";
    cout <<"the word The/the " <<endl;
    cout <<"When you are done, type in a '@' symbol" <<endl <<endl;
    cout <<"Let's get started...start typing in a pargraph!" <<endl <<endl;
}
```

Check to see if it is a Match

```
//check to see if the word is "the" or "The"
int check_word(char word[],char against[])
{
    uppercase(word);
    if (strcmp(word,against) == 0)
       return 1;
    return 0;
}
```

Our new Main function…

```
int main()
{
    char to_compare[MAX];       //word to compare to
    char aword[MAX];            //holds the current word
    int total = 0;              //holds the running total, starting at zero

    welcome();
    read_word(to_compare);  //read in a word to compare
    uppercase(to_compare);  //make sure it is uppercase
    do
    {       read_word(aword);   //get a word from the input buffer
            total += check_word(aword,to_compare);  //match?
    } while(!time_to_end());            //is it time to end?

    cout <<"There were " <<total <<" uses of the word " << to_compare<<endl;
    cin.get();
```

Graphical…User Interaction

- For our graphics example, let's write a program that will draw a word in the graphics window at any mouse click (left or right)
  - It will end when a middle mouse click is hit
- Algorithm:
  - Welcome the user…**explain the new rules**
  - **Read in the word to display**
  - Wait for a mouse hit
  - While it wasn't a middle button
    - Display the word at the x,y location of the hit

Getting Started….

```
#include <iostream>
using namespace std;
#include "graphics.h"

//This program is an example of user interaction using
//graphics.h
//written by Karla Fant

//Wait for a mouse button hit
//Find out the x,y location where the mouse click happened

const int LEFT = 513;  //code for the left button
const int RIGHT = 516; //code for the right button
const int MIDDLE = 519;//code for the middle button

bool get_mouse(int & x, int & y); //get the x,y location for any hit
void initialize();                //start up graphics
void welcome(char array[]);
```

Welcome Function

```
//Tell the user the rules
void welcome(char todisplay[])
{
    cout <<"We will be drawing interactively today a word or character ";
    cout <<"of your choice " <<endl;
    cout <<"Hit the middle button to end!" <<endl;
    cout <<endl;
    cout <<"What do you want to display? ";
    cin >>todisplay;    cin.get();
    cout << "Please select the location on the board with the mouse:\n ";
}
```

Initialize Graphics Window

```
void initialize()
{
     int window_size;

    cout << "Please select the size of your window: ";
    cin >> window_size;  cin.get();

    initwindow(window_size,window_size);

    setcolor(YELLOW);      //colors go from 0 through 15
    setlinestyle(0,0,6);       //Solid, No pattern, 6 is VERY wide
    settextstyle(0,0,5);      //create a really large character (5)
    settextjustify(1,1);       //center both horizontally and vertically
}
```

Get the Mouse Location upon click…

```
//Get the location of a mouse button hit...from any button
//Return false if the middle button is hit!
bool get_mouse(int & x, int & y)
{

   //wait for any mouse click at all

   while (!ismouseclick(LEFT) &&
         !ismouseclick(MIDDLE) && !ismouseclick(RIGHT));

   x = mousex();
   y = mousey();

   if (ismouseclick(MIDDLE))      //middle button happened
      return false;
   clearmouseclick(LEFT);         //important! Reset the mouse
   clearmouseclick(MIDDLE);
   clearmouseclick(RIGHT);
   return true;
}
```

The Main function…

```
int main()
{
    int x, y;                   //location for drawing
    char todisplay[21];

    initialize();
    welcome(todisplay);

    while(get_mouse(x,y))           //mouse click?
                outtextxy(x,y,todisplay);   //output text at the mouse click

    //pause
    getch(); //for graphics window
    cin.get(); //for console window
    return 0;
}
```

Graphical…User Interaction

- Now let's change it up!
- If the left button is hit, draw
- If the right button is hit, change the color

```
//Cycle through the colors!
void cyclecolor()
{
	int color = getcolor();
	color = (color + 1) % getmaxcolor();
	setcolor(color);
}
```

Get the Mouse Location upon click…

```
//Get the location of a mouse button hit...from any button
//Return false if the middle button is hit!
bool get_mouse(int & x, int & y)
{
   //wait for any mouse click at all
   while (!ismouseclick(LEFT) &&
         !ismouseclick(MIDDLE) && !ismouseclick(RIGHT));

   x = mousex();
   y = mousey();

   if (ismouseclick(MIDDLE))      //middle button happened
      return false;
   if (ismouseclick(RIGHT))       //change the color
      cyclecolor();

   clearmouseclick(LEFT);        //important!
   clearmouseclick(MIDDLE);      //important!
   clearmouseclick(RIGHT);       //important!
   return true;
}
```