Today in CS161

- **Week #8 Practicing!**

- **Writing Programs to Practice**
  - Write a program that counts the number of vowels in a sentence, ended by a period
  - Write a program that creates an advertisement for the Oregonian – taking out all vowels except those that start as the first character of a word
Counting Vowels – Solution #1

- Count the number of vowels…using arrays of characters…word by word…
- First, write the algorithm:
  - Prompt the user to enter in a sentence
  - Read a word
    - For every character in the word, if the character is an a, e, i, o or u increment a counter by 1.
    - Do this until the ‘\0’ is reached.
  - Display the results
Welcome...and Read a word...

//inform the user of the rules
void welcome()
{
    cout <<"Please enter a sentence - terminated by a period" <<endl;
    cout <<"When you are done hit enter" <<endl <<endl <<endl;
}

//Read in a word
void read_word(char aword[])
{
    cin.width(MAX); //make sure the word isn't too long!
    cin >>aword;    //skips leading whitespace and reads until whitespace
    cin.get();
}
Count Vowels

// Count the vowels in the word
int count_vowels(char array[])
{
    int length = strlen(array);       // find out how many characters to go thru
    int num_vowels = 0;
    for (int i = 0; i < length; ++i)
        if (array[i] == 'a' || array[i] == 'A' ||
            array[i] == 'e' || array[i] == 'E' ||
            array[i] == 'i' || array[i] == 'I' ||
            array[i] == 'o' || array[i] == 'O' ||
            array[i] == 'u' || array[i] == 'U') // It is a vowel!
            ++ num_vowels;  // add one to the vowel counter
    return num_vowels;
}
// Is it time to end? If the end of the word is a period....
bool time_to_end(char aword[])
{
    bool yes;
    int length = strlen(aword);
    if (aword[length-1] == '.')
        yes = true; // yep - end of the sentence!
    else
        yes = false;
    return yes;
}
int main()
{
    char word[MAX];       //it will hold the current word
    int vowels = 0;       //vowel counter

    welcome();
    do
    {
        read_word(word);   //read in a word
        vowels += count_vowels(word);  //keep track of # vowels
    } while (!time_to_end(word));

    cout <<"You entered: " <<vowels <<" Vowels!" <<endl;
    cin.get();
    return 0;
}
#include <iostream>
#include <cstring>

using namespace std;

//This program is written by Karla Fant to demonstrate
//the use of functions, arrays of characters, and the subscript operator
//to access individual elements of an array

void welcome();                        //describes the rules
int count_vowels(char word[]);        //counts the vowels in a word
void read_word(char word[]);          //reads in a word from the user
bool time_to_end(char word[]);        //does the word end in a period?

const int MAX = 21;                   //maximum array size for this program
Running the program

Please enter a sentence - terminated by a period
When you are done hit enter

This is a fun program that will
count the number of vowels.
You entered: 16 Vowels!
Counting Vowels – Solution #2

- Count the number of vowels...using arrays of characters...reading in the entire sentence
- First, write the algorithm:
  - Prompt the user to enter in a sentence
  - Read a sentence
    - For every character in the sentence, if the character is an a, e, i, o or u increment a counter by 1.
    - Do this until the ‘\0’ is reached.
  - Display the results
Main...simplified!

```cpp
int main()
{
    char array[MAX]; // it will hold the current sentence (131)
    int vowels = 0; // vowel counter

    welcome();
    read_sentence(array); // read it all in!
    vowels += count_vowels(array); // keep track of # vowels

    cout << "You entered: " << vowels << " Vowels!" " << endl;

    cin.get();
    return 0;
}
```
Read in a sentence.....

//Read in the entire sentence
//and ignore the ending period and newline that follows

void read_sentence(char array[])
{
    cin.get(array,MAX,'.');  //read in an entire sentence
    cin.ignore(100,'\n');   //ignore the period and newline afterwards
}

***nothing else changes! ***

***the function to count vowels remains exactly the same!!! ***
## Putting it all together.

```cpp
#include <iostream>
#include <cstring>

using namespace std;

//This program is written by Karla Fant to demonstrate
//the use of functions, arrays of characters, and the subscript operator
//to access individual elements of an array

void welcome(); //describes the rules
int count_vowels(char array[]); //counts the vowels in an array
void read_sentence(char array[]); //reads in an entire sentence

const int MAX = 131; //maximum number of characters in a sentence
```
Running the program

Please enter a sentence - terminated by a period
When you are done hit enter

This is a great program
that will keep reading until
131 characters or a period.
You entered: 24 Vowels!
If we wrote one more function

Let’s call it “isvowel”

We can re-use that function any time we are wondering if a character is a vowel.

```cpp
//Check to see if a particular character is a vowel
bool isvowel(char ch)
{
    //First let's lower case the character:
    ch = tolower(ch);

    if (ch == 'a' || ch == 'i' || ch == 'e' || ch == 'o' || ch == 'u')
        return true;
    return false; //not a vowel!
}
```
Count Vowels

//Count the vowels in the word
int count_vowels(char array[])
{
    int length = strlen(array); //find out how many characters to go thru
    int num_vowels = 0;
    for (int i = 0; i < length; ++i)
        if (isvowel(array[i])) // It is a vowel!
            ++ num_vowels; //add one to the vowel counter

    return num_vowels;
}

• The benefit is now we can use the “isvowel” function for other programs! Let’s see…
Creating an Advertisement...

- Our next program today is to create an advertisement in the want-ad’s. Since each line costs money, we will see what the ad is like if we take out all of the vowels.
- Of course, we don’t want to take out any vowels that are the first letter of a word…as those words would just not make sense.
- If the word is less than 4 characters, then all vowels stay...
  - Let’s think about what functions we will need…and the best way to start with that is to write an algorithm!
Creating an Advertisement...Algorithm

Algorithm...working word by word

- Welcome the user. Ask them to enter in a line for an advertisement. After each line they will be asked whether or not there will be another line
- Prompt the user to enter in the first line of the ad
- Read in a word
  - Find out the length of the word
  - If it is less than 4 characters, display it as is
  - Otherwise, display the first character of the word
  - For all of the rest of the characters in the word, display them ONLY if they are not a vowel
- Continue with the next word, until a newline is next in the input buffer
- Ask the user if they have another line. If so, continue reading and processing each word
Welcome...and Read a word...

//inform the user of the rules
void welcome()
{
    cout <<"Please enter the first line of your advertisement" <<endl;
    cout <<"The resulting ad will be displayed and you will be asked ";
    cout <<"if you have another line ...";
    cout <<"When you are done hit enter" <<endl <<endl <<endl;
}

//Read in a word
void read_word(char array[])
{
    cin.width(MAX); //make sure all words are within range
    cin >>array; //skip leading whitespace, read in characters
    //until whitespace is encountered but not read
}
//Display the word...using the rules outlined earlier
void display_word(char array[]) {
    int length = strlen(array);  //what is the length?

    if (length < 4)                //a short word...just display it!
        cout << array << ' ';
    else
        {
            cout << array[0];      //otherwise the first character is always
                                   //displayed
            for (int i=1; i<length; ++i)  //go through all characters in the word
                if (!isvowel(array[i])) //it is not a vowel
                    cout << array[i];   //so output the character
            cout << ' ';          //have a space occur after the word
        }
}
// Check to see if a particular character is a vowel
bool isvowel(char ch)
{
    // First let's lower case the character:
    ch = tolower(ch);

    if (ch == 'a' || ch == 'i' || ch == 'e' || ch == 'o' || ch == 'u')
        return true;
    return false;  // not a vowel!
}
Is it the end of a line?  Or play again?

//Find out if we are at the end of a line....
bool end_of_line()
{
    if (cin.get() == '\n') //we know there will be whitespace....
        return true; //we are at the end
    return false; //nope...not yet
}

//Does the user want to enter in another line?
bool again()
{
    char response; //holds the y or n entered by the user

    cout << endl << endl << "Would you like to enter another line? Y or N ";
    cin >> response; cin.get();

    if (response == 'y' || response == 'Y') //YES!!
        return true;
    return false;
}
Main...*think of this as the glue!*

```c
int main()
{
    char array[MAX];           //it will hold the current word
    welcome();
    do
    {
        do
        {
            read_word(array);    //read a word
            display_word(array);  //display the appropriate parts of the word
        } while (!end_of_line());  //continue for the rest of the line
    } while (again());          //does the user want to do this again?

    cin.get();
    return 0;
}
```
#include <iostream>
#include <cstring>
#include <cctype>
using namespace std;

//This program is written by Karla Fant to demonstrate how we can read in an array and output only select elements of the array. This program creates an advertisement where the vowels are stripped away -- the exceptions are when the vowel is located as the first element or if the word is short (less than 4 characters)

void welcome(); //describes the rules
void read_word(char word[]); //read a word
void display_word(char word[]); //display the word without vowels
bool isvowel(char); //is the character a vowel?
bool end_of_line(); //Did we reach the end of the line?
bool again(); //does the user want to enter another?
Running the program

Please enter the first line of your advertisement
The resulting ad will be displayed
and you will be asked if you have another line ...When you are done hit enter

$1290 3 bedroom, 2 bath ranch style detached home in Beaverton
$1290 3 bdrm, 2 bth rnch styl dtchd hm in Burtn

Would you like to enter another line? Y or N Y

Dishwasher, Garage Door Opener, Fenced Backyard
Dshwshr, Grg Dr Opnr, Fncd Bckyrd

Would you like to enter another line? Y or N Y

All utilities are included
All utlts are inclld

Would you like to enter another line? Y or N N
Today in CS161

Next Topic: Practicing!

Writing Programs to Practice

- Write a game program (1 player) of Mad Math
- Reuse the functions to provide for multiple players

- Rewrite the same program using “structures” to group together related topics
  - Greatly simplifying the ability to have multiple players!
Mad Math – One Player

- A game that displays an equation and the player must come up with the correct answer. As their score increases, so does their level
- First, write the algorithm:
  - Describe the rules
  - Get the users name and capitalize each word
  - Play the game
    - Display an equation
    - Get the answer
    - Check to see if the answer is correct
    - Increase/decrease the points
    - Display the points
    - Continue until the user is done
//describe this game to the user
void welcome()
{
    cout <<"Welcome to the Mad about Math program\n\n";
    cout <<"The goal is to get as many equations correct\n";
    cout <<"You get 1 point for each correct answer and -2 for each wrong!";
    <<endl <<endl;

    cout <<endl <<endl <<"Let's begin " <<endl <<endl;
    srand(time(0));
}
void get_name(char name[]) 
{
    cout <<"What is your name? ";
    cin.get(name,MAX);
    cin.ignore(100,\n');
    capitalize(name);       //make sure each word is capitalized
}
//Capitalize the first letter of each word in the name
void capitalize(char name[]) 
{
    int length = strlen(name);
    name[0] = toupper(name[0]); //capitalize the first character of the name

    //Find the blanks in a name
    for (int i=0; i< length; ++i)
    {
        if (name[i] == ' ') //the next character needs to be capitalized
            name[i+1] = toupper(name[i+1]);
    }
}
Show an equation

//Play the game! The argument indicates the complexity of the numbers
int equate(int max)
{
    int first;  //first number
    int second;  //second number
    int operation;  //type of operation
    int answer;   //answer supplied by user
    int correct;  //correct answer

    first = rand() % max;
    second = rand() % max;
    operation = rand() % 4;
if (operation == 0)  //addition
    {
        cout <<"SOLVE: " <<first <<" + " <<second <<" Equals: ";
        correct = first + second;
    }
else if (operation == 1)  //subtraction
    {
        cout <<"SOLVE: " <<first <<" - " <<second <<" Equals: ";
        correct = first - second;
    }
else if (operation == 2)  //multiplication
    {
        cout <<"SOLVE: " <<first <<" x " <<second <<" Equals: ";
        correct = first * second;
    }
else  //division
    {
        cout <<"SOLVE: " <<first <<" / " <<second <<" Equals: ";
        correct = first / second;
    }
Show an equation...continued

```cpp
    cin >> answer;

    if (answer == correct)
        return 1; // correct answer
    return -2; // incorrect answer
}

// Does the user want to play again?
bool play_again()
{
    char answer;
    cout << "Do you want to play again? ";
    cin >> answer; cin.get();
    return (answer == 'y' || answer == 'Y');
}
```
int main()
{
    int level = 10; //simplest level
    char player[MAX];
    int points = 0; //player's points
    welcome();
    get_name(player);
    do
    {
        //show the equation
        points += equate(level);
        show_score(player, points);
        progress(level, points); //should they progress a level?
    }
    while (play_again());
    ending_message(player, points);
    cin.get(); return 0;
}
Running the program

Welcome to the Mad about Math program

The goal is to get as many equations correct.
You get 1 point for each correct answer and -2 for each wrong!

What is your name? sally van dyke

Let's begin Sally Van Dyke

$\text{SOLVE: } 6 \div 1 \text{ Equals: } 6$
Sally Van Dyke's score is: 1

Do you want to play again? y
$\text{SOLVE: } 9 - 1 \text{ Equals: } 8$
Sally Van Dyke's score is: 2

Do you want to play again? y
$\text{SOLVE: } 0 \times 3 \text{ Equals: } 0$
Sally Van Dyke's score is: 3

Do you want to play again? y
$\text{SOLVE: } 4 \div 4 \text{ Equals: } 1$
Sally Van Dyke's score is: 4

Do you want to play again? y
$\text{SOLVE: } 9 - 2 \text{ Equals: } 7$
Sally Van Dyke's score is: 5

Do you want to play again? y
$\text{SOLVE: } 3 + 8 \text{ Equals: } 11$
Sally Van Dyke's score is: 6

Do you want to play again? y
$\text{SOLVE: } 4 \div 3 \text{ Equals: } 1$
Sally Van Dyke's score is: 7

Do you want to play again? y
$\text{SOLVE: } 10 \times 13 \text{ Equals: } 130$
Sally Van Dyke's score is: 8

Do you want to play again? y
$\text{SOLVE: } 28 - 5 \text{ Equals: } 23$
Sally Van Dyke's score is: 9

Do you want to play again? n

Wonderful Game!
Sally Van Dyke's score is: 9
If we have written the functions for the single player problem well enough, we can simply reuse them for the next player...let's see:

```cpp
int play_game(char player[], int & points, int & level)
{
    cout <<player <<"'s turn: ";
    points += equate(level);
    show_score(player,points);

    progress(level,points); //should they progress a level?
}
```
# Main...with two players

```c
int main()
{
    int player1_level = 10;  //simplest level
    int player2_level = 10;

    char player1[MAX];      //player1
    char player2[MAX];      //player2
    int player1_points = 0;
    int player2_points = 0;

    welcome();
    cout << "First player: ";
    get_name(player1);      //get the names of the two players
    cout << "Next player: ";
    get_name(player2);
}
```
Main...with two players

do
{
    //let each player do an equation...
    play_game(player1, player1_points, player1_level);
    play_game(player2, player2_points, player2_level);
}

if (player1_level == player2_level && player1_points == player2_points)
    cout << "GREAT JOB! You are BOTH winners today "
        << "with " << player1_points << " points" << endl << endl;
else if (player1_level > player2_level || player1_points > player2_points)
    cout << "THE WINNER IS: " << player1
        << "with " << player1_points << " points" << endl << endl;
else
    cout << "THE WINNER IS: " << player2 << " with "
        << player2_points << " points" << endl << endl;
Running the program

The goal is to get as many equations correct
You get 1 point for each correct answer and -2 for each wrong!

Let’s begin

First player: What is your name? sam smith
Next player: What is your name? sally miller
Sam Smith’s turn: SOLVE: 2 x 7 Equals: 14
Sam Smith’s score is: 1

Sally Miller’s turn: SOLVE: 0 + 9 Equals: 9
Sally Miller’s score is: 1

Do you want to play again? y
Sam Smith’s turn: SOLVE: 7 + 9 Equals: 16
Sam Smith’s score is: 2

Sally Miller’s turn: SOLVE: 9 + 6 Equals: 15
Sally Miller’s score is: 2

Do you want to play again? n
GREAT JOB! You are BOTH winners today with 2 points
What is a Structure

- Using structures (a new concept we will use in CS162), we can simplify this further and easily allow additional players.
- Think about what a player is…a player has a name, a score, and a level.
- A structure is a way for us to group different types of data together under a common name.
- With an array, we are limited to having only a single type of data for each element...
  - We’d need an array of players names
  - Another for the players scores
  - Another for the players levels. Too complicated!
What is a Structure

- With a structure, on the other hand, we can group each of these under a common heading
  - So, each player can now have a name, score, and level tied to it
  - And, we can then generalize this to allow for an array of players...and add as many as we want!
Why would we use a Structure

- Some people argue that with C++ we no longer need to use the concept of structures.
- And, yes, you can do everything that we will be doing with structures, with a “class” (which we learn about next term!)
- My suggestion is to use structures whenever you want to group different types of data together, to help organize your data.
How do you define a Structure?

- We typically define structures “globally”
  - this means they are placed outside of the main
- We do this because structures are like a “specification” or a new “data type”
  - which means that we would want all of our functions to have access to this way to group data, and not just limit it to some function by defining it to be local
How do you define a Structure?

- Each component of a structure is called a member and is referenced by a member name (identifier).
- Structures differ from arrays in that members of a structure do not have to be of the same type. And, structure members are not referenced using an index.
How do you define members of a Structure?

- A structure might look like:

```c
struct storeitem {
    char item[20];
    float cost;
    float price;
    int barcode;
};
```

- In this example, item, price, cost and barcode are member names. storeitem is the name of a new derived data type consisting of a character array, two real numbers, and an integer.
How do you define members of a Structure?

A structure might look like:

```c
struct player   //a player is: name, a score, and a level
{
    char name[MAX];
    int points;
    int level;
};     //<---- notice the semicolon!
```
How do you define instances of a Structure?

- Once you have declared this new derived data type, you can create instances -- variables (or “objects”) which are of this type (just like we are used to):

```
player player1;
```

Or, create an array:
```
player all_players[100];
```
How do you define instances of a Structure?

- By saying:
  ```
  player player1;
  ```
  - From this statement, `player1` is the variable (or object)
  - It has a name, score (#points) and level.
  - Just think of `player` as being a type of data which consists of an array of characters, two integers in this case.
How do you access members of a Structure?

- By saying:
  
  ```
  player player1;
  ```

- To access a member of a structure variable, we use a dot (the “direct member access” operator) after the structure variable’s identifier:
  
  ```
  player1.name  is the array of characters
  ```

  ```
  player1.points  is the integer
  ```

  ```
  player1.level  is the level
  ```
How do you access members of a Structure?

- We can work with these members in just the same way that we work with variables of a fundamental type:

  - To read in a name, we can say:
    ```
    cin >> player1.name
    Or, cin.get(player1.name, 21);
    ```

  - To display the score, we say:
    ```
    cout << player1.points
    ```
What operations can be performed?

- Just like with arrays, there are very few operations that can be performed on a complete structure.
- We can’t read in an entire structure at one time, or write an entire structure, or use any of the arithmetic operations...
- We can use assignment, to do a “memberwise copy” copying each member from one struct variable to another.
How do you define arrays of Structures?

- But, for structures to be meaningful when representing a deck cards, a store inventory, or a number of players for a game.
  - we may want to use an array of structures
  - where every element represents a different player in the game…
How do you pass Structures to functions?

- To pass a structure to a function, we must decide whether we want pass by reference or pass by value.
- By reference, we can pass 1 player:
  
  ```c
  returnType function(player & arg);
  ```

  // or an array of players:
  ```c
  returnType function(player arg[]);
  ```
#include <iostream>
using namespace std;

// This program simulates a "mad about math" game
// written by Karla Fant for CS161 demonstrations

const int MAX = 21;
const int NUM = 2;

struct player { // a player has a name, a score, and a level
    char name[MAX];
    int points;
    int level;
}; //<---- notice the semicolon!
This is the same as before....

void welcome(char name[]);        // display the rules
void capitalize (char name[]);    // capitalize each word in a name
// capitalize the first character of each word in a name

void capitalize(char name[])
{
    int length = strlen(name);
    name[0] = toupper(name[0]); // capitalize the first character of the name

    // Find the blanks in a name
    for (int i=0; i< length; ++i)
        if (name[i] == ' ') // the next character needs to be capitalized
            name[i+1] = toupper(name[i+1]);
}
This is the same as before....

```cpp
void get_name(char name[]) {
    cout << "What is your name? ";
    cin.get(name, MAX);
    cin.ignore(100, '\n');
    capitalize(name); // make sure each word is capitalized
}

// describe this game to the user
void welcome() {
    cout << "Welcome to the Mad about Math program\n\n";
    cout << "The goal is to get as many equations correct\n";
    cout << "You get 1 point for each correct answer and -2 for each wrong!"
        << endl << endl;
    cout << endl << endl << "Let's begin " << endl << endl;
    srand(time(0));
}
```
New Stuff...using the struct!

```c++
int play_game(player & a_player)
{
    //Tell which player’s turn it is…
    cout << a_player.name << "'s turn: ";

    //Give the player an equation to calculate
    a_player.points += equate(a_player.level);

    //Show their score
    show_score(a_player.name, a_player.points);

    //should they progress a level?
    progress(a_player.level, a_player.points);
}
```
int main()
{
    player players[NUM]; //we have two players

    welcome();

    //initialize the level and points
    for (int i = 0; i < NUM; ++i)
    {
        players[i].level = 10;
        players[i].points = 0;
        cout <<"For player # " <<i+1 <<": ";
        get_name(players[i].name);
    }
}
Main...now using the structure!

//Time to play the game!
do{
    for (int i = 0; i < NUM; ++i) //let each player do an equation...
        play_game(players[i]);
} while (play_again());

winning_message(players);

    cin.get();
return 0;
Display the winning message...

```java
void winning_message(player all[])
{
    int highest_score = -99;
    int highest_index = -99;
    bool found_tie = false;

    //Find the highest score
    for (int i = 0; i < NUM; ++i)
    {
        //find the player with the highest score
        if (all[i].points > highest_score)
        {
            highest_score = all[i].points;
            highest_index = i;
        }
    }
}
```
Display the winning message...

// Now see if there is a tie
for (int i = 0; i < NUM; ++i)
{
    // Find the player with the highest score
    if (all[i].points == highest_score && i != highest_index)
    {
        found_tie = true;
        cout << "GREAT JOB! We have a tie today! "
            << all[i].name << " and " << all[highest_index].name << " have "
            << highest_score << " points" << endl << endl;
    }
}
if (!found_tie) // There was not a tie
{
    cout << "THE WINNER IS: " << all[highest_index].name
        << " with " << highest_score << " points" << endl << endl;
}
Welcome to the Mad about Math program

The goal is to get as many equations correct
You get 1 point for each correct answer and -2 for each wrong!

Let's begin

For player # 1: What is your name? kent worth
For player # 2: What is your name? sue smith
Kent Worth's turn: SOLVE: 4 / 8 Equals: 0
Kent Worth's score is: 1

Sue Smith's turn: SOLVE: 2 + 9 Equals: 2
Sue Smith's score is: -2

Do you want to play again? y
Kent Worth's turn: SOLVE: 9 + 0 Equals: 9
Kent Worth's score is: 2

Sue Smith's turn: SOLVE: 6 + 9 Equals: 1
Sue Smith's score is: -4

Do you want to play again? n

THE WINNER IS: Kent Worth with 2 points
Mad Math Game...Many Players!

- With structures and arrays of structures we can make a slight jump with minimal code modifications to allow for many players!

```cpp
#include <iostream>
using namespace std;

//This program simulates a "mad about math" game
//written by Karla Fant for CS161 demonstrations

const int MAX = 21;
const int NUM = 5;

***THE ONLY CHANGE***
Running the program

Welcome to the Mad about Math program

The goal is to get as many equations correct
You get 1 point for each correct answer and -2 for each wrong!

Let’s begin

For player # 1: What is your name? sam smith
For player # 2: What is your name? sue miller
For player # 3: What is your name? beth adams
For player # 4: What is your name? tyler van adams
For player # 5: What is your name? keith williams

Sam Smith’s turn: SOLVE: 7 / 3 Equals: 2
Sam Smith’s score is: 1

Sue Miller’s turn: SOLVE: 2 / 4 Equals: 9
Sue Miller’s score is: -2

Beth Adams’s turn: SOLVE: 4 x 4 Equals: 16
Beth Adams’s score is: 1

Tyler Van Adams’s turn: SOLVE: 7 + 1 Equals: 8
Tyler Van Adams’s score is: 1

Keith Williams’s turn: SOLVE: 6 - 2 Equals: 1
Keith Williams’s score is: -4

Do you want to play again? n

THE WINNER IS: Sam Smith with 2 points