

# Discussion 4!

GDB, arrays, OOP

Note: have CAEN logged in and ready for this discussion!



GDB

# Using GDB

Extremely helpful for finding logic errors in your code!

Allows you to walk through your code and see what variables are at any point

What is a breakpoint?

# How to run GDB

On CAEN:

Compile your code by running:

```
g++ -std=c++98 -g -Wall programFile.cpp -o programFile.exe
```

Then run:

```
gdb programFile.exe
```

# Common GDB Commands

Useful Commands: [https://umich.instructure.com/files/21281857/download?download\\_frd=1](https://umich.instructure.com/files/21281857/download?download_frd=1)

run : runs your code until a breakpoint is hit

b <sourceFile: lineNumber> : sets a breakpoint at that line number

b <functionName>: sets a breakpoint at the function

p <variableName>: prints the current value of the variable specified

s : This command will execute the next statement, but if the statement is a function, “s” will step into the function and stop at the first executable statement within the function

n: Same as “s”, but won’t step into any functions

c : This command will continue execution of your program from the current statement until the program ends or a break point is reached.

# GDB Demo

```
4  int findMax(int arr[], int size) {  
5      int max;  
6  
7      for(int i = 0; i < size; ++i) {  
8          if(arr[i] > max) {  
9              max = arr[i];  
10         }  
11     }  
12  
13     return max;  
14 }
```

# Run the code

- Run “git clone [https://github.com/emolson16/gdb\\_practice](https://github.com/emolson16/gdb_practice)” in CAEN
- cd gdb\_practice
- Compile and run gdbDemo.cpp using the instructions in the program
  - g++ -std=c++98 -g -Wall GDBpractice.cpp -o GDBpractice.exe



# Arrays



# General syntax

```
int myArray[25];
```



type



name



size

```
int myArray[5] = {2, 4, 6, 8, 10};
```



initialization

1. What's stored in the array?

# General syntax

```
int myArray[25];
```



type



name



size

```
int myArray[5] = {2, 4, 6, 8, 10};
```



initialization

1. What's stored in the array?

“Memory Junk”

# General syntax

```
int myArray[25];
```



type



name



size

```
int myArray[5] = {2, 4, 6, 8, 10};
```



initialization

1. What's stored in the array?  
"Memory Junk"
2. Why?

# General syntax

```
int myArray[25];
```



type



name



size

```
int myArray[5] = {2, 4, 6, 8, 10};
```



initialization

1. What's stored in the array?

“Memory Junk”

2. Why?

It's faster

# Accessing elements

```
int myArray[5] = {1, 3, 5, 7, 9};
```

```
myArray[2] = 7;
```

```
myArray[5] = 12;
```



What does this line of code do?

# Accessing elements

```
int myArray[5] = {1, 3, 5, 7, 9};
```

```
myArray[2] = 7;
```

```
myArray[5] = 12;
```



What does this line of code do?

# Why do we use arrays?

They are extremely efficient with memory and time use!

# A note on passing arrays into functions

You must specify what type of variable you are passing into any function

```
int max(arr[]);
```

```
int max(char arr[], int n);
```

```
int max(string arr[], int n);
```



# Passing in 2d Arrays

You must specify the type and number of columns

```
7  
8 void printArray(char arr[][8], int rows) {  
9 |
```

# Why can't you pass a whole matrix in?

Because you aren't just passing a "matrix," you're passing an array of arrays

So your array type is certain size of array

```
void printAry(int rows,
              const char ary[][2])
{
    int i;
    int j;
    for (i = 0; i < rows; i++)
    {
        for (j = 0; j < 2; j++)
        {
            cout << ary[i][j] << " ";
        }
        cout << endl;
    }
}
```



# Some array questions

# Example

Given an array of 'size' and an int, return a boolean for whether or not the int is found in the array

```
5    bool isFound(int arr[], int size, int num);  
6  
7    int main(){  
8  
9        int arr[5] = {1, 2, 3, 4, 5};  
10  
11        cout << isFound(arr, 5, 6);  
12  
13        return 0;  
14    }
```

# Example Solution

Given an array of 'size' and an int, return a boolean for whether or not the int is found in the array

```
5    bool isFound(int arr[], int size, int num){
6        for(int i = 0; i < size; ++i){
7            if(arr[i] == num){
8                return true;
9            }
10       }
11       return false;
12   }
```

# Example- 2d arrays

Create a function that finds the maximum element of a 2d array of ints with 5 columns

The parameters are the 2d array and the number of rows

# Example Solution

```
4  int findMax(int arr[][5], int rows) {  
5      int max = arr[0][0];  
6  
7      for(int i = 0; i < rows; ++i) {  
8          for(int j = 0; j < 5; ++j) {  
9  
10             if(arr[i][j] > max) {  
11                 max = arr[i][j];  
12             }  
13  
14         }  
15     }  
16     return max;  
17 }
```



# Object Oriented Programming



# Class vs. Object

A class is a template for objects

Classes describe object behavior

Each object is an instance of a class

Class

Object

```
2  using namespace std;
3  #include <iostream>
4
5  class Cup {
6      private:
7          int ounces;
8          string color;
9
10     public:
11         void fill(int addedOunces);
12
13         bool isEmpty();
14
15     };
16
17     int main(){
18
19         Cup soloCup;
20
21         return 0;
22     }
```

# Private vs. Public

**By default, classes are *private***

You cannot access private variables or functions from outside of the class

You can access anything from inside the class

```
5  class Cup {
6      int ounces;
7      string color;
8
9      void fill(int addedOunces);
10
11     bool isEmpty();
12
13 };
14
15 int main(){
16
17     Cup soloCup;
18
19     soloCup.fill();
20
21     return 0;
22 }
```

# Private vs. Public

By default, classes are *private*

**You cannot access private variables or functions from outside of the class**

You can access anything from inside the class

```
2  using namespace std;
3  #include <iostream>
4
5  class Cup {
6      private:
7          int ounces;
8          string color;
9
10     public:
11         void fill(int addedOunces);
12
13         bool isEmpty();
14
15 };
16
17 int main(){
18
19     Cup soloCup;
20
21     cout << soloCup.ounces << endl;
22
23     return 0;
24 }
```

# Private vs. Public

By default, classes are *private*

You cannot access private variables or functions from outside of the class

**You can access anything from inside the class**

```
5  class Cup {  
6      private:  
7          int ounces;  
8          string color;  
9  
10     public:  
11         void fill(int addedOunces){  
12             ounces += addedOunces;  
13         }  
14  
15         bool isEmpty();  
16  
17     };  
18  
19     int main(){  
20  
21         Cup soloCup;  
22  
23  
24         return 0;  
25     }  
26
```

# Using classes

To declare an instance of a class: **MyClassName myInstance**

To access attributes: **myInstance.attributeName**

To call methods: **myInstance.methodName()**

```
BankAccount myAccount;  
myAccount.deposit(5);
```

# Example

CPP file:

<https://drive.google.com/file/d/1Fxlui-2A4mTRKe0Mg2yghcFUXRBtiw1/view?usp=sharing>

.txt version:

<https://drive.google.com/file/d/1AwQv3RE5sgBKvzNLmTK-yRLogqo2iCVu/view?usp=sharing>

Git:

\$ git clone <https://github.com/emolson/16/ooop-example.git>

```
5 // this is a class!
6 class BankAccount {
7     private:
8         double bill = 100; //represents credit card bill
9         double balance = 0; // represents your balance
10
11
12     public:
13
14         //TODO: deposit the given amount into your balance
15         void deposit(double amount) {
16
17         }
18
19         //TODO if you have enough money, withdraw the given amount and return true
20         // if you don't have enough money, just return false
21         bool withdraw(double amount) {
22
23         }
24
25         //Pay your credit card bill (you can go into debt here)
26         void payBill();
27
28         // Challenge problem- don't worry if you can't get it yet
29         // TODO pay your friend the given amount to their account
30         // You cannot go into debt here (return false if you don't have enough money)
31         bool payFriend(BankAccount& friendAccount, double amount) {
32
33         }
34
35         // prints current balance
36         void printBalance(){
37             cout << "Current balance is: $" << balance << endl;
38         }
39     };
40
41     //TODO pay your credit card bill (you can go into debt here)
42     void BankAccount::payBill(){
43
44     }
45 }
```

# Example Solution

```
6 class BankAccount {
7     private:
8         double bill = 100; //represents credit card bill
9         double balance = 0; // represents your balance
10
11
12     public:
13
14         //TODO: deposit the given amount into your balance
15         void deposit(double amount) {
16             balance += amount;
17         }
18
19         //TODO if you have enough money, withdraw the given amount and return true
20         // if you don't have enough money, just return false
21         bool withdraw(double amount) {
22
23             if(balance >= amount) {
24                 balance -= amount;
25                 return true;
26             }
27
28             return false;
29         }
30
31         //Pay your credit card bill (you can go into debt here)
32         void payBill() {
33             balance -= bill;
34             bill = 0;
35         }
36
37         // Challenge problem- don't worry if you can't get it yet
38         // TODO pay your friend the given amount to their account
39         // You cannot go into debt here (return false if you don't have enough money)
40         bool payFriend(BankAccount& friendAccount, double amount) {
41
42             if(balance >= amount) {
43                 balance -= amount;
44                 friendAccount.deposit(amount);
45             }
46
47             return false;
48         }
49
50         // prints current balance
51         void printBalance(){
52             cout << "Current balance is: $" << balance << endl;
53         }
54     };
55 }
```

## Using the BankAccount class...

```
48  int main() {  
49  
50      BankAccount myAccount;  
51      myAccount.deposit(100);  
52  
53      BankAccount friendAccount;  
54  
55      myAccount.payFriend(friendAccount, 75);  
56  
57      myAccount.printBalance();  
58      friendAccount.printBalance();  
59  
60      return 0;  
61  }
```

What would this print?



## Using the BankAccount class...

```
48  int main() {  
49  
50      BankAccount myAccount;  
51      myAccount.deposit(100);  
52  
53      BankAccount friendAccount;  
54  
55      myAccount.payFriend(friendAccount, 75);  
56  
57      myAccount.printBalance();  
58      friendAccount.printBalance();  
59  
60      return 0;  
61  }
```

What would this print?

Current balance is \$25

Current balance is \$75