Discussion 4!

GDB, arrays, OOP

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



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

run: runs your code until a breakpoint is hit

b <souceFile: 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.

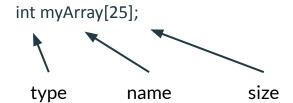
GDB Demo

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

Run the code

- Run "git clone https://github.com/emolson16/gdb_practice" in CAEN
- cd gdb_practice
- Compile and run gdbDemo.cpp using the instructions in the program
 - o g++ -std=c++98 -g -Wall GDBpractice.cpp -o GDBpractice.exe

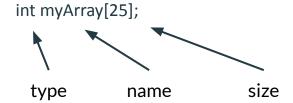
Arrays



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



1. What's stored in the array?

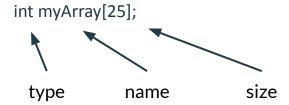


int myArray $[5] = \{2, 4, 6, 8, 10\};$



1. What's stored in the array?

"Memory Junk"



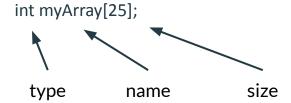
int myArray $[5] = \{2, 4, 6, 8, 10\};$



1. What's stored in the array?

"Memory Junk"

2. Why?



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



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;

What does this line of code do?

myArray[5] = 12;
```

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

```
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

```
bool isFound(int arr[], int size, int num);
 6
     int main(){
 8
          int arr[5] = \{1, 2, 3, 4, 5\};
 9
10
11
          cout << isFound(arr, 5, 6);</pre>
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

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

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

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

```
using namespace std;
                          #include <iostream>
                          class Cup {
                              private:
                                  int ounces;
                                  string color;
      Class
                              public:
                    11
                                  void fill(int addedOunces);
                    12
                                  bool isEmpty();
                          };
                          int main(){
Object
                              Cup soloCup;
                    19
                    21
                              return 0;
```

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

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

```
using namespace std;
#include <iostream>
class Cup {
    private:
        int ounces;
        string color;
    public:
        void fill(int addedOunces);
        bool isEmpty();
};
int main(){
    Cup soloCup;
    cout << soloCup.ounces << endl;</pre>
    return 0;
```

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

```
class Cup {
         private:
             int ounces;
             string color;
         public:
             void fill(int addedOunces){
11
12
                 ounces += addedOunces;
             bool isEmpty();
17
     };
     int main(){
         Cup soloCup;
         return 0;
```

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/1FxvIui-2A4mTRKe0Mg2yghcFUXRBtiw1/view?u sp=sharing

.txt version:

https://drive.google.com/file/d/1AwQv3 RE5sgBKvzNLmTK-yRLogqo2iCVu/view?u sp=sharing

Git:

\$ git clone https://github.com/emolson 16/oop-example.git

```
class BankAccount {
   double bill = 100; //represents credit card bill
   double balance = 0; // represents your balance
   //TODO: deposit the given amount into your balance
   void deposit(double amount) {
   bool withdraw(double amount) {
    //Pay your credit card bill (you can go into debt here)
   void payBill();
   // Challenge problem- don't worry if you can't get it yet
   // TODO pay your freind the given amount to their account
    // You cannot go into debt here (return false if you don't have enough money)
   bool payFriend(BankAccount& friendAccount, double amount) {
   void printBalance(){
     cout << "Current balance is: $" << balance << endl;</pre>
//TODO pay your credit card bill (you can go into debt here)
void BankAccount::payBill(){
```

Example Solution

```
class BankAccount {
   double bill = 100; //represents credit card bill
   double balance = 0; // represents your balance
   void deposit(double amount) {
     balance += amount;
   bool withdraw(double amount) {
      if(balance >= amount) {
       balance -= amount:
    void payBill() {
     balance -= bill;
     bill = 0:
   bool payFriend(BankAccount& friendAccount, double amount) {
      if(balance >= amount) {
       balance -= amount;
       friendAccount.deposit(amount);
   void printBalance(){
     cout << "Current balance is: $" << balance << endl;</pre>
```

Using the BankAccount class...

```
int main() {
49
       BankAccount myAccount;
51
       myAccount.deposit(100);
52
53
       BankAccount friendAccount;
54
       myAccount.payFriend(friendAccount, 75);
57
       myAccount.printBalance();
       friendAccount.printBalance();
       return 0;
61
```

What would this print?

Using the BankAccount class...

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

What would this print?

Current balance is \$25 Current balance is \$75