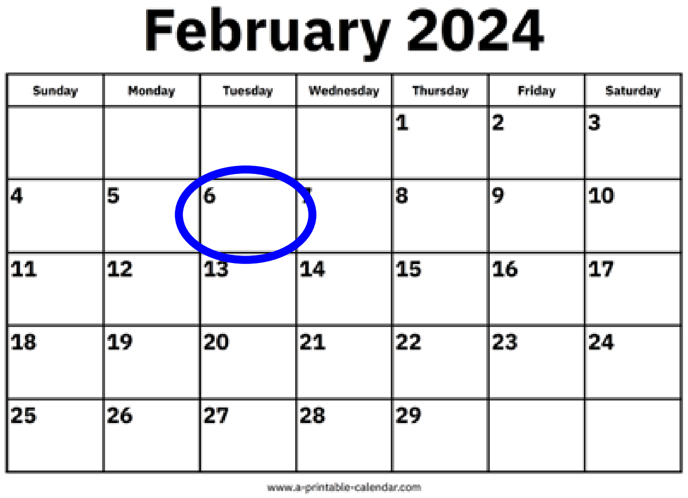
CS 211 : Tues 02/06 (lecture 10)

Prof. Hummel
(he/him)

- **Topics**: classes, objects, object-oriented programming (OOP) in C++



February 2024

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29		

www.a-printable-calendar.com

Notes:

- *Lecture slides available on Canvas*
- ***Project 04** can still be submitted tonight, 2 parts:*
  - *ram.c*
  - *tests.c*
- ***HW 05** (intro to C++) next Tuesday 2/13*
- ***Extra-credit C project** will be released tonight*

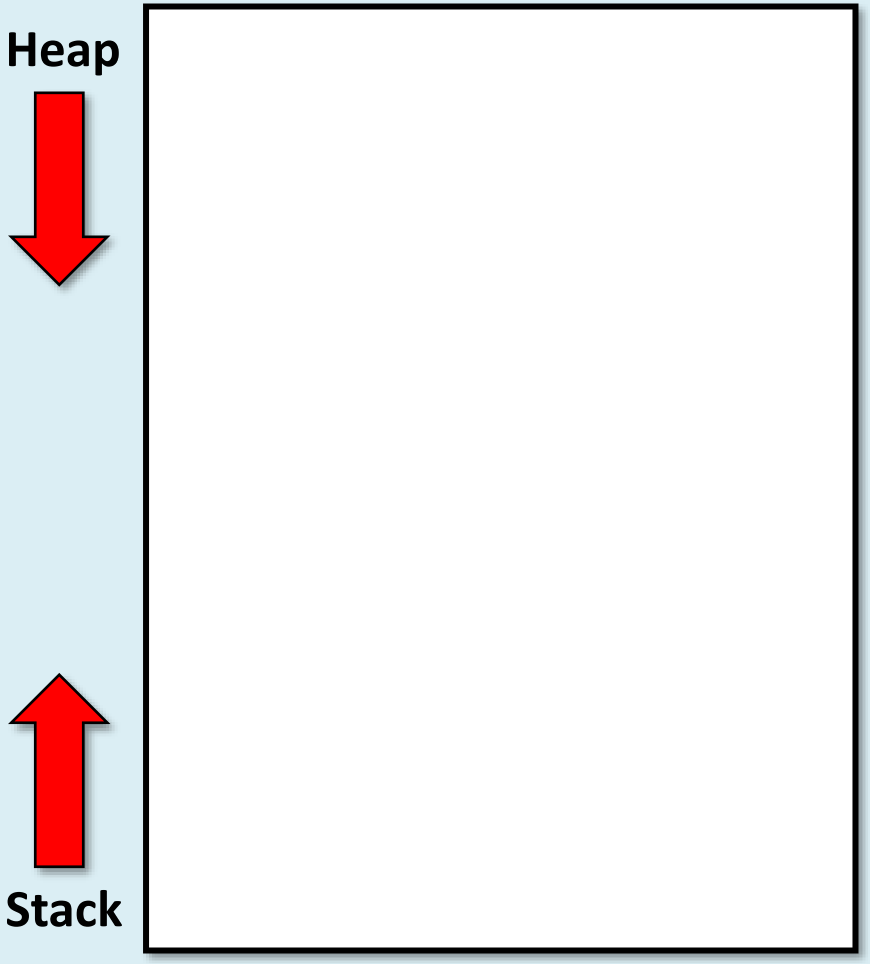
Northwestern University

Project 04

- **Dynamically-allocated array…**

```
struct RAM
{
  struct RAM_CELL* cells;
  int     num_values;
  int     capacity;
};
```

```
TEST(memory, write_one)
{
  struct RAM* memory;

  memory = ram_init();
```



Example of an "off by one" error

```
1  bool ram_write_cell_by_addr(struct RAM* memory, struct RAM_VALUE value, int address)
2  {
3    if (address<0 ||
4        address>memory->num_values) {
5      return false;
6    }
7    else if (!memory->cells[address].identifier) {
8      return false;
9    }
10
11   if(memory->cells[address].value.value_type == RAM_TYPE_STR){
12     free(memory->cells[address].value.types.s);
13   }
14
15   memory->cells[address].value = value;
16
17   if (value.value_type == RAM_TYPE_STR) {
18     memory->cells[address].value.types.s = strdup(value.types.s);
19   }
20
21   return true;
22 }
```

A) Line 3

B) Line 4

C) Line 7

D) Line 15

E) Line 18

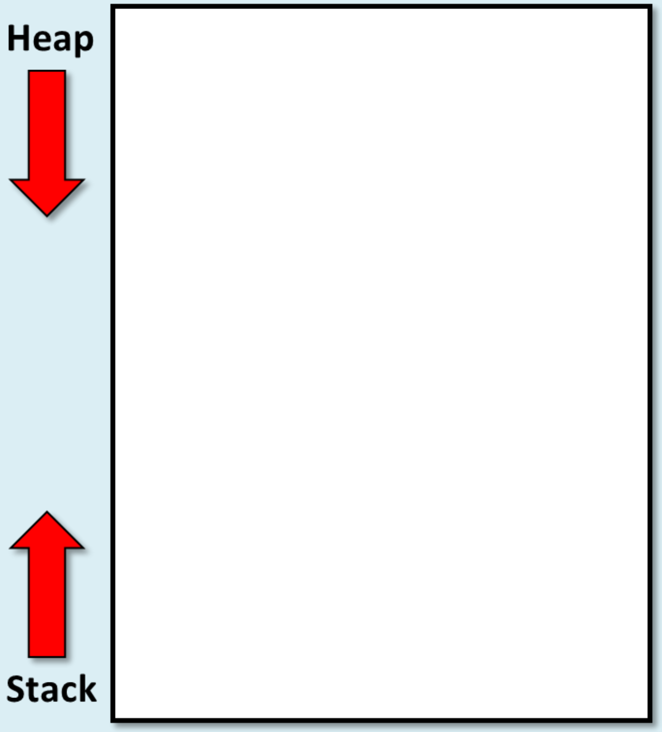
Writing test cases

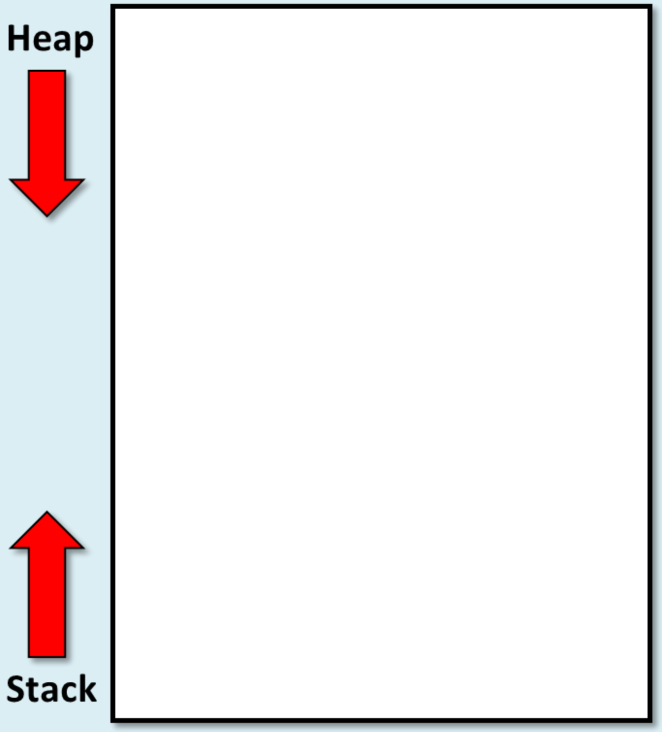
```
TEST(memory, grow)
{
    .
    .
    .
    bool success = ram_write_cell_by_id(memory, s, "x");
    success = ram_write_cell_by_id(memory, s, "y");
    success = ram_write_cell_by_id(memory, k, "z");
    success = ram_write_cell_by_id(memory, b, "a");

    success = ram_write_cell_by_id(memory, s, "b");
    success = ram_write_cell_by_id(memory, s, "c");
    success = ram_write_cell_by_id(memory, k, "d");
    success = ram_write_cell_by_id(memory, b, "e");

    success = ram_write_cell_by_id(memory, s, "f");
    success = ram_write_cell_by_id(memory, s, "g");
    success = ram_write_cell_by_id(memory, k, "h");
    success = ram_write_cell_by_id(memory, b, "i");

    success = ram_write_cell_by_id(memory, s, "j");
    success = ram_write_cell_by_id(memory, s, "k");
    success = ram_write_cell_by_id(memory, k, "l");
    success = ram_write_cell_by_id(memory, b, "m");
    ASSERT_TRUE(success);

    ASSERT_TRUE(memory->num_values == 16);
    ASSERT_TRUE(memory->capacity == 16);

    success = ram_write_cell_by_id(memory, b, "n");
    ASSERT_TRUE(memory->num_values == 17);
    ASSERT_TRUE(memory->capacity == 32);

```

This is a good start for testing that memory grows properly.

What feedback would you give for improving the test?

```
TEST(memory, write_one)
{
  struct RAM* M;
  M = ram_init();

  struct RAM_VALUE v;
  v.value_type = RAM_TYPE_INT;
  v.types.i    = 11;

  char id[6];
  strcpy(id, "x");

  ram_write_cell_by_id(M, v, id);
```





- **Why learn different programming languages?**
- **Languages influence how you think…**

Examples

- **Racket?**
  - *Functional (no variables, functions, mathematical foundation)*
- **Python?**
  - *Scripting (easier to program, doesn't scale to large apps)*
- **C?**
  - *Low-level, imperative (memory/ptrs), procedural (functions)*
- **C++?**
  - *Higher-level, imperative,* ***object-oriented***

If we moved Project 04 to C++...

```
struct RAM
{
  struct RAM_CELL* cells;
  int      num_values;
  int      capacity;
};
```

```
TEST(memory, write_one)
{
  struct RAM* M;
  M = ram_init();

  struct RAM_VALUE v;
  v.value_type = RAM_TYPE_INT;
  v.types.i     = 11;

  ram_write_cell_by_id(M, v, "x");

  ram_destroy(M);
}
```

data members

constructor

destructor

function members ("methods")

```
class RAM
{
private:
    struct RAM_CELL* cells;
    int      num_values;
    int      capacity;

public:
    RAM();
    ~RAM();

    write_cell_by_id(...);
    .
    .
    .
};
```

```
TEST(memory, write_one)
{
   RAM  M;
   RAM_VALUE v;
   v.value_type = RAM_TYPE_INT;
   v.types.i     = 11;

   M.write_cell_by_id(v, "x");
}
```

Why OOP?

- Classes package data and code together…
- Classes model reality / closer to how we think…

```
class Sailboat
{
public:
  string Name;
  double LengthOverall;
  double LengthWaterline;

  Sailboat(string name,
           double length,
           double lwl);

  double maxSpeedKts();
  double maxSpeedMPH();
};
```

Sailboats

- **A programming example**

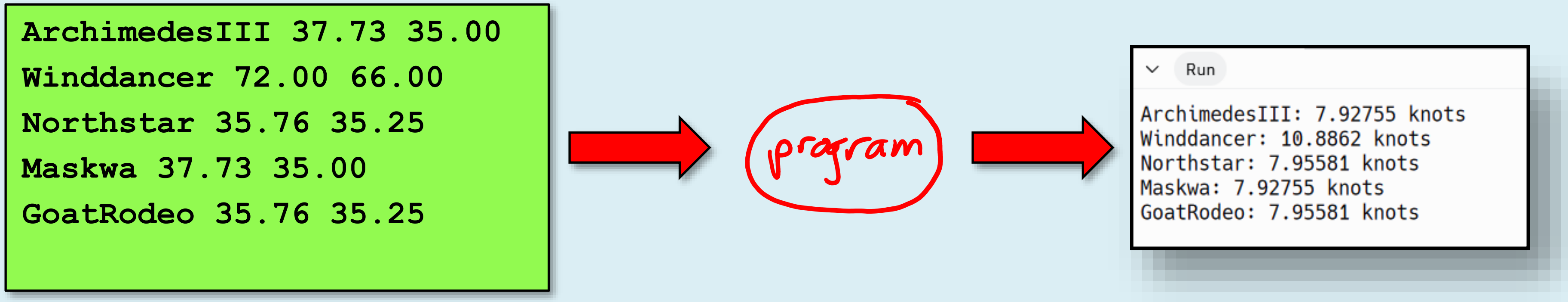
displacement

foiling

Max speed = 1.3 * $\sqrt{\text{length}}$

Demo

- Login to replit.com
- Open team…
- Open project "Lecture 10"



```
ArchimedesIII 37.73 35.00
Winddancer 72.00 66.00
Northstar 35.76 35.25
Maskwa 37.73 35.00
GoatRodeo 35.76 35.25
```

```
ArchimedesIII: 7.92755 knots
Winddancer: 10.8862 knots
Northstar: 7.95581 knots
Maskwa: 7.92755 knots
GoatRodeo: 7.95581 knots
```

sailboat.h / sailboat.cpp

```
/*sailboat.h*/

#pragma once

#include <string>
using namespace std;

class Sailboat
{
public:
  string Name;
  double LengthOverall;
  double LengthWaterline;

  Sailboat(string name,
           double length,
           double lwl);

  double maxSpeedKts();
  double maxSpeedMPH();
};
```

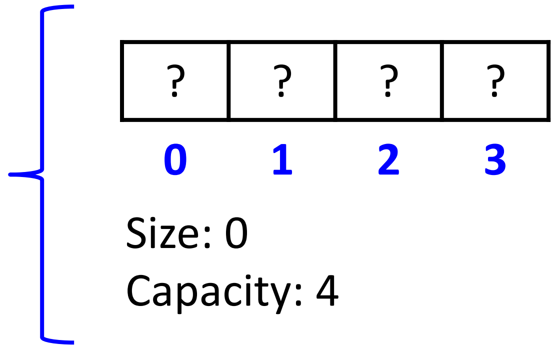
```
/*sailboat.cpp*/
#include <cmath>
#include "sailboat.h"
using namespace std;

Sailboat::Sailboat(string name, double length,
                   double lwl) {
  this->Name = name;
  this->LengthOverall = length;
  this->LengthWaterline = lwl;
}

// given a boat, return its max speed in knots
double Sailboat::maxSpeedKts() {
  return 1.34 * sqrt(this->LengthWaterline);
}

// given a boat, return its max speed in MPH
double Sailboat::maxSpeedMPH() {
  return 1.1 * this->maxSpeedKts();
}
```

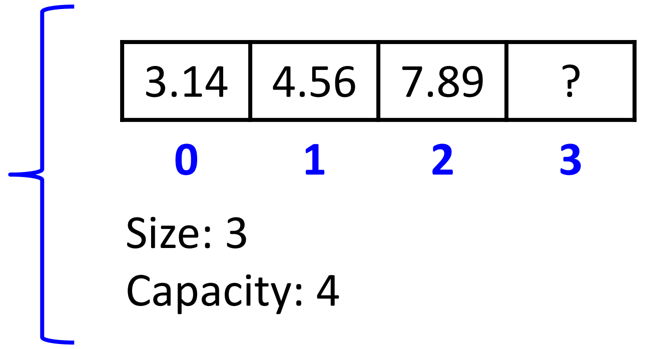
vector<T>

- **A one-dimensional container that resizes as needed**

```
#include <vector>
using namespace std;
```

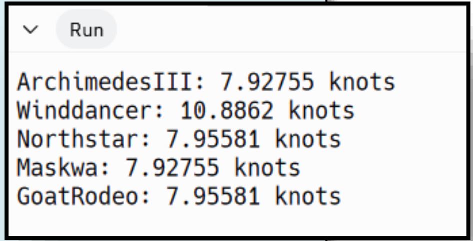
```
vector<double>  V;




V.push_back(3.14);
V.push_back(4.56);
V.push_back(7.89);
```





main.cpp

```cpp
/*main.cpp*/

#include <iostream>
#include <fstream>
#include <vector>
#include <string>

#include "sailboat.h"

using namespace std;

vector<Sailboat> readBoats(string filename)
{ … }

int main()
{
  vector<Sailboat> boats = readBoats("boats.txt");

  if (boats.size() == 0) {
    cout << "No sailboat data..." << endl;
    return 0;
  }

  for (Sailboat s : boats)
    cout << s.Name << ": " << s.maxSpeedKts() << " knots" << endl;

  return 0;
}
```



```
ArchimedesIII: 7.92755 knots
Winddancer: 10.8862 knots
Northstar: 7.95581 knots
Maskwa: 7.92755 knots
GoatRodeo: 7.95581 knots
```

Data hiding

- **A good practice is to hide implementation details**
  - *Prevents access / unnecessary errors*
  - *Allows those details to change if necessary*

```cpp
class Sailboat
{
public:
   string Name;
   double LengthOverall;
   double LengthWaterline;

   Sailboat(string name,
            double length,
            double lwl);

   double maxSpeedKts();
   double maxSpeedMPH();
};
```

```cpp
class Sailboat
{
private:
   string Name;
   double LengthOverall;
   double LengthWaterline;

public:
   Sailboat(string name,
            double length,
            double lwl);

   double maxSpeedKts();
   double maxSpeedMPH();

   string getName();
   double getLength();
   double getLengthWaterline();

};
```

accessors ("getters")

What should I be working on?

Finish ***project 04*** *if not already…*

HW 05 *(intro C++) next Tuesday…*