EEL 4837
Programming for Electrical Engineers II

Ivan Ruchkin
Assistant Professor
Department of Electrical and Computer Engineering
University of Florida at Gainesville
iruchkin@ece.ufl.edu
http://ivan.ece.ufl.edu

Course Overview

Readings:

- The syllabus (see Canvas for the latest version)

About the Course

- Programming foundations specifically for Electrical Engineers
- Implementation and use of data structures in C++
- Algorithm design principles
- "Excursions" exploring the role of algorithms in Electrical Engineering applications

Writing efficient program requires understanding of the underlying foundations:

- Manipulation and analysis of structured data
- Understanding how algorithms are built
- Trade-offs between program performance and resource constraints.

Why Take this Course?

- Algorithm analysis provides a means to distinguish between **what is practically possible** and **what is practically impossible**
- Efficient algorithms lead to efficient programs that make better use of computer resources
- Efficient programs sell better
- Programmers who write efficient programs are preferred

This is a programming course, with supporting mathematics to enable understanding of computation and efficiency

Who should take this course?

- You want to have a grasp of fundamentals of data structures and algorithms
- You want to be able to assess the impact of various programming decisions on performance
- You want to have understanding of how some of the tools you use are built (e.g., SPICE)
- You want the option of applying for a job that includes proficiency with software

Course content at glance

- **Data Structures:**
  - Arrays, Matrices, Stacks, Queues, Trees, Graphs, Priority Queues, Maps
  - Heaps, Binary Search Trees, AVL trees, Directed Acyclic Graphs (DAGs)
- **Algorithms:**
  - Searching, Sorting, Expression Evaluation, Topological Sort, Graph Search (breadth/depth-first search), Spanning Tree
  - Dijkstra, Prim, Kruskal
- **Algorithm Design Principles:**
  - Recursion, Divide-and-Conquer, Greedy, Dynamic Programming
- **Programming and Implementation:**
  - C++ features, Templates, Pointers, Memory Management
- **Complexity Foundations:**
  - P, NP, NP-Completeness, Computability

Instructor

- **Ivan Ruchkin**
  - **Office:** LAR 334B (behind the wooden doors on Larsen 3rd floor)
  - **Office Hours:** Tuesdays 11:30am-12:30pm (after the lecture)
  - **Email:** iruchkin@ece.ufl.edu (I prefer contact via Canvas, unless it's an emergency)
- **Research interests:**
  - **Cyber-Physical Systems, Formal Methods, AI/ML**
  - **Trustworthy Engineered Autonomy (TEA) Lab**
  - **Current research projects (details at http://ivan.ece.ufl.edu)**
    - Guarantees for systems with learning components
    - Quantification of confidence in safety (e.g., absence of collisions)
    - Autonomous car racing
- **Past Experience**
  - Several years as a programmer in enterprise systems, desktop applications, and open-source software
  - PhD in Software Engineering
  - Research experience in specification, verification, monitoring, and assurance for cyber-physical systems

Teaching Assistant

Qiangeng Yang

Email: q.yang@ufl.edu (Canvas preferred)

Office hours: Friday 4p–6p, NEB 401 (flexible hours, fill out the weekly poll)

- Please contact us through Canvas and copy all of us. If you send emails to our email address we may not able to respond in time
- For common questions, we will use either the Discussions or Piazza (TBD from the survey)
- Pay attention to the **Canvas Calendar**. It always has a current view of the scheduled course activities (lectures, office hours, deadlines)

Course Materials

Textbooks

- Mark Allen Weiss: Data Structures and Algorithms in C++ 4th Edition, Addison-Wesley **(Required)**
- Ellis Horowitz, Sartaj Sahni, Susan Anderson-Freed. Fundamentals of Data Structures in C, 2nd Edition, W. H. Freeman **(Recommended)**
- Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms 3rd Edition, MIT Press **(Optional)**

This is a University course. You are expected to explore the topic beyond lectures, go through the requisite chapters of the book, look for additional materials in the Web, etc.

C++ books

C++ Programing Textbook

- C++ How to Program, 10th Edition by Paul Deitel and Harvey Deitel.

Advanced Textbook

- The C++ Programming Language, 4th Edition by Bjarne Stroustrup
  - Only for experienced C++ programmers

Evaluation and Grading

Assignment	Percentage of Final Grade
Homeworks (4)	40%
Excursions (2)	20%
Midterm Exam	20%
Final Exam	20%

- **Midterm:** March 9 (Thursday before Spring Break), in person in class (10:40a–12:35p)
- **Final:** May 4 (online, via Canvas+HonorLock)

Final Grading Scale

Percent	Grade	Grade Points
93.0 - 100	A	4.00
90.0 – 92.9	A-	3.67
87.0 – 89.9	B+	3.33
83.0 – 86.9	B	3.00
80.0 – 82.9	B-	2.67
77.0 – 79.9	C+	2.33
73.0 – 76.9	C	2.00
70.0 – 72.9	C-	1.67
67.0 – 69.9	D+	1.33
63.0 – 66.9	D	1.00
60.0 – 62.9	D-	0.67
0 – 59.9	E	0.00

- No curving down: everyone can get an A
- May curve up particularly challenging assignments
- Probably no overall curve at the end

What You Should Know Before the Course

- Experience in writing programs (e.g., EEL 3834)
- The course will use C++ but I do not expect you to have background in C++ coming in. However, you should some experience coding in a high-level language (e.g., Python or Java).
- We will explain some ideas using code snippets in C++ (and sometimes pseudocode)
- Your homeworks will include programming assignments in C++
- **Fill out the survey by end of Wednesday (tomorrow):** https://forms.gle/TtbT7gC5EfAKipoq8

Expectations from You

- Attend lectures (be on time)
- Take notes whenever necessary
- Come to class prepared!
- Ask questions, participate in class
- For exams, study the corresponding chapters of textbooks and beyond in addition to class lectures
- The load will be moderately heavy. Be prepared to work hard!
- Be honest to yourselves

Expectations from Us

- Interesting and relevant material
- Post slides before the lecture
- Lecture recordings will be posted (but PLEASE attend!)
- Practice problem sets to help prepare for exams
- Help in office hours & online (within 48 hours)
- Fair grading
- Timely feedback within 1 week

More on Homework Assignments

- **Do them to understand the material, not just get a grade**
- **Content from lectures and readings**
- **All homework writeups & code must be your own work**
  - Peer discussions about the problem in general are encouraged
  - Copying each others' work or directly from the Web is strictly forbidden!
  - There is a fine line between plagiarism and collaboration
- **Due due ~10 days after release (Thursday -> Sunday 11:59p)**
- **Late homework submissions will be penalized 10%/day**
  - But the homework with lowest grade will be dropped.

More on Excursions

- **Larger projects grounded in EE applications**
- **Planning on 2 excursions**
  - First: early February -> early March
  - Second: mid-March -> mid-April
- **Requires substantial implementation; led by the TA**
- **Submit only your individual work, same as with homeworks**
- **Late excursion submissions will be penalized 10%/day**

Cheating and Academic Dishonesty

- **Absolutely no form of cheating will be tolerated**
- You are all adults and we will treat you so
- See syllabus, UF Policy, and UF Honor code
- Cheating 0 for the assignment + referral to UF Honor Code Process (no exceptions)

Algorithms and Data Structures

- Algorithm: a step-by-step procedure for solving a problem in a finite amount of time. For example: how to find somebody's telephone in a telephone book.
- Data structure: a way data is organised in computer memory; for example: array, list, list of lists, tree, table...

C++ Overview

Readings:

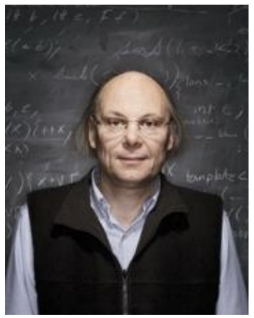
- Stroustrup 2.2
- Deitel chapters 2–5

Quick C++ Overview

You should get used to some basics of C++ like this code:

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

int main() {
   cout << "Hello World!" << endl;
   return 0;
}
```



C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows away your whole leg.

--Bjarne Stroustrup

C/C++ Types

Basic data types:

- Character (char) : upper/lower-case letters, numbers, special characters, *etc.* [256 ASCII Characters]
- Integer (int)
- Float (float)
- Double (double)
- void

Type	Size (bits)	Size (bytes)	Range
char	8	1	-128 to 127
unsigned char	8	1	0 to 255
int	16	2	-2^{15} to $2^{15}-1$
unsigned int	16	2	0 to $2^{16}-1$
short int	8	1	-128 to 127
unsigned short int	8	1	0 to 255
long int	32	4	-2^{31} to $2^{31}-1$
unsigned long int	32	4	0 to $2^{32}-1$
float	32	4	3.4E-38 to 3.4E+38
double	64	8	1.7E-308 to 1.7E+308
long double	80	10	3.4E-4932 to 1.1E+4932

Type conversion:

- Implicit/ promotion:

```
int a = 5; float b;
b = 12.0 + a; // 17.0
```

- Explicit/ casting:

```
int a = 5, b = 2; double c;
c = (double) a/b; // 2.5
```

C/C++ Operators

In order of precedence:

- Post increment/decrement: `a = 10; b = 5; a = a + (b++);`
- Pre increment/decrement: `a = 10; b = 5; a = a + (++b);`
- sizeof [compile time operator to get byte size of variable or data type], ! [logical not]
- *, /, %
- +, -
- <, <=, >, >=
- ==, !=
- && [logical AND]
- || [logical OR]
- Conditional/ternary operator: (?:) `outcome = (score > 42) ? ‘W’ : ‘L’;`
- Assignments: =, +=, *=, -=, /=, %=

C/C++ Control Flow

- **Conditional statements:**

```
if (A) {
    //code executes if A holds
} else if (B) {
    // code runs if “B and not A”
} else {
    // code runs if “not B and not A”
}
```

- **While loops:**

```
// while loop from 1 to 5
int i = 1;
while (i <= 5) {
    cout << i << " ";
    ++i;
}
```

- **Do-while loops:**

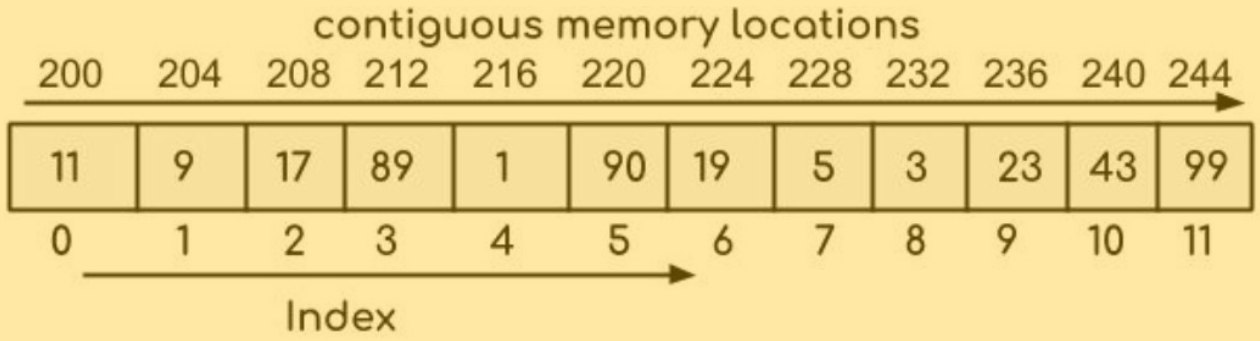
```
int i = 1;
// do...while loop from 1 to 5
do {
   cout << i << " ";
   ++i;
} while (i <= 5);
```

- **For loops:**

```
for (int i = 0; i < 5; i++) {
     cout << i << "\n";
}
```

Array Data Structure

The [one-dimensional] array is a data structure consisting of a collection of *elements,* each identified by at least one *array index* or *key.* The keys are generally chosen as a contiguous sequence of numbers.



Very close to (but a little different from) the underlying memory layout.

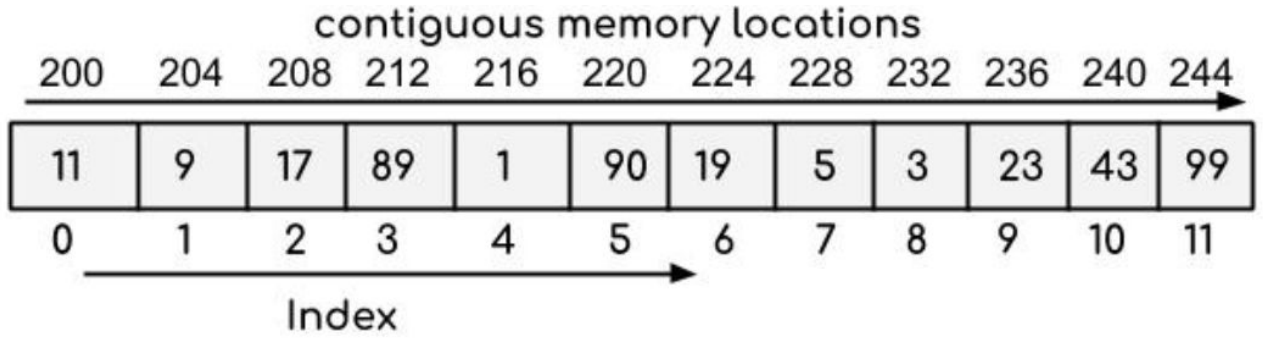
Data structure with a few primitive operations:

```
CreateArray(n)   // creates array of size n (conceptual)
a[i] = x         // assigns the value x at index i
x = a[i]         // reads the value of a[i] to x
```

Algorithm Example: Linear Search

```
int lSearch(int arr[], int n, int x)
{
    int i;
    for (i = 0; i < n; i++)
        if (arr[i] == x)
            return i;
    return -1;
}

void main()
{
    int arr[] = { 3, 4, 1, 7, 5 };
    int n = 5;
    int x = 4;
    int index = lSearch(arr, n, x);
    if (index == -1)
        cout << "Element is not present in the array” << endl;
    else
        cout << "Element found at position " << index << endl;
}
```



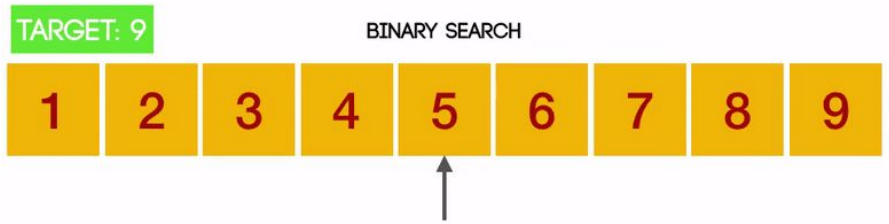
Is the number 26 in the array?

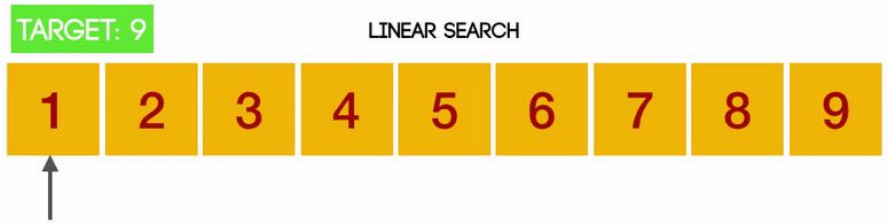
Searching a Sorted Array

2	5	10	12	15	20	25	31	40
0	1	2	3	4	5	6	7	8

Is the number 26 in the array?

```
int binarySearch(int arr[], int left, int right, int x) {
  while (left <= right) {
      int mid = left + (right - left) / 2;
      if (arr[mid] == x)  return mid;
      else if (arr[mid] < x) left = mid + 1;
            else  right = mid - 1;
 }
 return -1;
}
```





Is binary search a more efficient algorithm than linear search?

Data Structures as C++ Classes

```
Class Circle {
   private:
     double radius;
   public:
     Circle () {radius = 0.0;}
     Circle (double r);
     double setRadius (double r);
     double getArea ();
     double getCircumference ();
}
Circle::Circle (double r) {
 radius = r;
}
double Circle::getArea () {
  return radius * radius * (22.0/7);
}
double Circle::getCircumference () {
  return 2 * radius * (22.0/7);
}
```

```
void main () {
    Circle c (7);

    cout << "The area of circle c is"
         << c.getArea()
         << endl;
}
```

Compilation in C++

- Executable code is stored in source (.cpp) files
- Header (.h and .hpp) files store declarations of classes/functions
- Making an executable from code is a **three-step process**:
  - Preprocessing, compilation, linking
  - For most of this course, you'll be able to ignore this (Deitel 1.9, 1.10)
- The **compiler command** is g++
- *Example:*
  - Compile: g++ helloworld.cpp -o hello
  - Run: ./hello
- Use any **IDEs/Text editors**: *VS Code, Sublime, Eclipse, Notepad++*