Table of Content

Overview

Computer Science skill should go far beyond just programming itself. It should focus on problems solving ability with computational thinking even for grade schools. Automation is entrenched in our daily lives in the era of the digital age. Computer Science with computational thinking is indispensable for strengthening the foundation.

Storming Robots utilizes Robotics to animate problem-solving effort starting in Grade 4. However, we encourage students to study in this Algorithms in C/C++ Track starting from Grade 8.

This syllabus consists of four levels. The levels will be comparable up to college level from Freshman to junior year topics in CS, such as data structure, introduction to algorithms with combinatorial optimization, and complexity analysis.

Characteristics (important to read first)

1) **Focus on problems solving**/software development skill, so students will not work with the physical robot.
2) **Progress is self-paced**. All assigned exercises should be completed with excellent quality. In the later levels, more emphasize in robustness, memory consumption, readability, maintainability, etc. All levels stress in Computational Thinking and Efficiency.
3) Students will be allowed to move quickly to the next concept after they demonstrate satisfactory level of understanding through their homework and/on pop-quiz in class if necessary. Those with gaps in their prerequisite knowledge will receive additional exercises to address the shortcomings.
4) Most exercises/instructions constantly require cognitive thinking and analysis. (i.e. no spoon feeding method).

Requirement for Level Promotion:

1) Only those who demonstrate satisfactory level of understanding will be promoted.
2) MUST note that just being able to show completed homework may not mean good level of understanding in certain concepts. The common cause of this symptom is substantial external assistance to complete their assigned work. Because of that, students miss out the critical process of analysis on their own. In this case, pop-quiz may be given for these students to "complete in class" in order to prove their understanding before promotion to the next level is allowed .

What was deemed to be "complete" with an assignment?

- Proper testing is required to ensure correctness .
- Any error must be corrected.
- Suggested improvement must be made if code is found poorly written and inefficient.
- Written material should be of the similar quality as what a professional would write; especially for the level III+.

How to do Well in this Track?

1) Manage your time wisely. Schedule time for homework proactively.
2) Review the book materials even after concepts are explained to you in class.
3) Completion of all exercises naturally follows deep learning, but not necessarily vice versa. It takes "self-disciplines" and "stay inquisitive".
   - Be self-disciplined — while there are a lot of learning materials online, such as stackOverflow, you should never just copy and paste. Acquire the "thinking process". This goes true as well if you receive external help.
   - Stay inquisitive— not just know how to implement the solutions, but to acquire the process of analysis to arrive the solutions.
   - Ask questions. Do not allow yourself to be satisfied just because a program set is complete, but only after you feel you do understand.
4) Should expect approx. 3+ hours of programming homework per week.
5) Must be perseverant in taking on challenging problems.

Aligning with Storming Robots Roadmap

Allow students to embed competitions and standardized exams in-between levels—

1) Students completed level I with high "proficiency" have the capacity to self-study for Advanced Placement Computer Science (AP CS A) . Majority of these students score 5 in AP with a small extent of review on fundamental object-oriented structure.
2) Many Level II students participate the USA Computing Olympiad (USACO) online exam as performance gauges. Do note USACO is far more demanding in terms of analytical, and computational programming skill than AP CS.
3) Students who achieved level II with high efficiency may even self-study the rest of the levels listed in this syllabus.

Learning Resources—Books and online packets:

Level I, II :

- C Programming: A Modern Approach, **2nd Edition** by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503.
- W3 resource exercise (optional) and https://learn.stormingrobots.com — Computer Science Track.

Level III :

- Part of K.N. King book

Level IV:

- Mastering Algorithms with C: ISBN-13: 978-1565924536, or ISBN-10: 1565924533

Level V:

- Data Structures and Algorithm Analysis in C++, 4th Edition by Mark A. Weiss—ISBN-10: 0273769383
- (optional) : C++ Primer (5th Edition) by y Stanley B. Lippman, Josée Lajoie, and Barbara E. Moo. ISBN-10: 0321714113

Software required

- Microsoft Visual Studio Community Version (Windows OS only). OR
- Visual Studio Code—runs on Linux, Windows, and MacOS.
  - Special note: We shall not be able to provide support any other IDE other than these two.

Honor Code

Programming assignments are pledged work and are bound by the honor code. To simply put, the violation may be in any of the following forms:

1. Claim someone else's work as their own.
2. Edit someone else's work by simply changing variables or style, etc., and claim to be the result of your own work.
3. Complete assigned work with substantial help from others to the extent that you cannot even explain your own work.

If found honor code is violated, parents will be informed. Students will not be allowed to participate in any competitions with Storming Robots. Repeating offenders will not be accepted back to SR programs.

Level B—I

The Fundamentals

This covers the bare fundamentals of computer programming from basic expressions to compound control structure.

Completion of Level B is required for all students who wish to participate in taking Electronic with Robotics at Storming Robots.

Learning Outcome

1) By the time this level is completed, students should have completed 20+/- programs up to Chapter 8 from the K.N. King book. This is not a hard-set number because more exercises will be given if necessary in order to strengthen one's understanding in a certain subject matter; or vice versa.
2) Possess the knowledge in the fundamentals of programming with an emphasis on producing clear, robust, and reasonably efficient code using top-down design, informal analysis, and effective testing and debugging.
3) Mindset in computational thinking and analysis.
4) Know how to use Debugger for trouble shooting.
5) Basic mechanic understanding in utilizing Array structure.
6) Exploratory usage of functions.
7) Control Structure:
   - For Level B — students should be able to do simple control structure, take a very *straight forward* problem set to resolve it with computer programming in C.
   - For Level I—students should be able to master nested control structure, analyze a problem set and figure out how to resolve it with computer programming in C.
8) Students demonstrated proficiency in level I will be able to self-study for AP Computer Science—A with great ease.

If you join this class with prior programming background in RobotC Robotics Projects I & II Analytics, it will help you to accelerate through the first five chapters.

Book :

C Programming: A Modern Approach, **2nd Edition** by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503. What will be covered

The Fundamentals

Covered Concepts:

1) How to use the Microsoft Visual C/C++ Compiler IDE.
2) C Fundamentals in writing Simple Program - Chapter 2
3) Standard Formatted Input/Output - Chapter 3
4) Base Conversions— Octet, Hexadecimal, Binary. (online note)
5) Functions — void, primitive data types return & simple pass in arguments.
6) Expressions (simple to compound) - Chapter 4
7) Rudimentary level in using functions() – class note
8) Selection/ if - Control Statements | Boolean Expressions - Chapter 5
9) Level I only—Introduction of State Tables/Diagrams. Simple idea in State machine—more example exercises using "switch" to implement state tables. More in depth will be covered in Later Levels.
10) Loops Control Structure - Chapter 6
11) For the following topics: Level B will cover only the mechanics of the structure. Level I will go more in-depth in the analytics manner.
    - Chapter 7 + and online note
    - Basic understanding in Array Type. Will focus only on the mechanic in using array—Chapter 8.
    - Rudimentary level of understanding in using struct. (online note)
12) Level I only—Multiple files within a project. (Partial Chapter 9 & 10).

All final work should follow prescribed Programming Style. There will be pop quiz given at random time to some students in order to ensure satisfactory proficiency in the covered concepts.

To conclude Level B :

Must finish up to point (11) listed above.

To conclude Level I :

Must finish up to point (12) listed above. For students who accomplish this level with high proficiency and wish to self-study AP CS—A, they should read up on fundamentals in object-oriented principle as how to create class and objects, such as using the AP Comp Sci Barron Book. Or, take an AP CS at SR during the summer.

Level II

Data Abstraction Type (ADT) - Linear Structure -1

This level involves software development skill well beyond Advanced Placement CS-A. Students should expect to work lesser number of exercises, but each require more time than previous level.

Learning Outcome:

1) Should be able to tackle Bronze in the USACO. Based on our history, all students who demonstrate high proficiency in completing Level II and have done some practices have promoted to at least Silver Level. Some even got Gold.
2) You will be stronger in programming analysis. Most programs exercises you have completed in even Level II under this program requires higher analytical skill over what is required in AP Comp. Sci.
3) Become more efficiency in Debugger including break points, observation of variables, watch feature.
4) Ability to recognize patterns and pay attention to reduction for efficiency instead of brute-force.
5) Should be able to conduct more complex project that typical College Freshman year level in CS.
6) Should feel comfortable to navigate around Linux System, and be able to build program and libraries yourself with gcc and g++.
7) Get experience in versioning control system—Git.

What will be covered

1) More advanced analytical work on Arrays
   a. Heavily focus robust implementation and error checking
   b. More focus on Multi- Dimensional
   c. Additional analytical work with nested loop .
2) Create functions & process from compilation to linkage - Chapter 9 & 10 (partial).
3) Bitwise operations — Chapter 20
4) Memory pointers with 1D and 2D array – Chapter 11 & 12 (Utilize the memory addresses shown in the debugger)
5) String manipulation — Chapter 13
   a. C string functions with focus on pointers arithmetic.
   b. array of pointers vs static 2D array,

c. Dynamic allocation—malloc/free vs realloc
d. Utilize memxxx() functions
e. How to utilize the memory addresses shown in the debugger
f. Command line arguments

6) Basics in Recursions (with class note)
   a. stack data structure — more in-depth understanding how it works in memory stack.
   b. Basics in dynamic programming method.
7) Commonly used Search / Sort algorithms :
   a. Basic in Binary Search (implementation)
   b. Insertion sort (implementation)
   c. Quicksort (Divide and Conquer Algorithm). (using the API)
8) Basic File I/O — Chapter 22 (with C & C++ - add'l note)
   a. Against multiple tests.
   b. Write programs to automate tests.
9) Learn to use Linux system (via Windows Linux SubSystemWSL) .
   a. Compile and link programs without using the IDE
   b. Automate testing with a set of test data files
   c. Create your own library.
   d. Basics in using Versioning Control System—Git.
10) Preprocessor - Chapter 14 —Conditional Compilation, such as #if, #elif, #line, etc. Its role in compilation process.
11) Work on two Bronze and two Silver level USACO problems

To conclude Level II:

- This level involves more mini-projects and exercises external to the book.
- Summary of the chapters covered in this level (not necessary in the this order): Chapter 9,to 16, and 20, 22.
- Reference : Standard Library—Ch. 21, 24

Level III

ADT - Linear Structure –2 & Intro to Non-Linear Structure

Completion of Level III is required for all students who wish to participate in any **Open level** of Robotics Competition.

Majority of exercises in this level are external from the book. Many of them take a more pragmatic approach to emphasize its application and analysis, and require much beyond just knowing the mechanic of coding.

Learning Outcome:

1) Complete College level Linear Data Structure.
2) Advanced understanding in Advanced memory pointers.
3) Know how to do generic programming
4) More seasoned in recursion and understanding basic dynamic programming.
5) Good understanding in Backtracking Algorithm with BFS for Maze Navigation (explore Tree and Graph structure).
6) Improvement in modular programming design, including abstraction layers.
7) By the time you complete this level, you are already equipped with very sound programming skill comparable to most undergrad CS capacity. Now, you are truly stepping into Computer Science — as opposed to programming — such as abstraction, correctness, complexity, and modularity. write code using external libraries when given a library interface.
8) During November and April, students are encouraged to continue working on USACO problems set, as well as take the Bronze to Silver Level online Exam.

Higher Expectation in the following:

1) Error Checking
2) Well tested
3) Concise and elegant , and well-documented
4) No program memory leaks

What will be covered

1) Struct, Union, Enumeration — Chapter 16.
   a. Using bits in structure
   b. Review Exercises to summarize Usage of array of struct, enum, and

pointers.

c. Big vs. Little Endian

d. Data Padding and Alignment

2) Advanced Uses of Pointers — Chapter 17 and class note

1. Polymorphism in C
2. Double pointer, Pointer to Pointer
3. Pointer to function, such as callbacks like in Quick Sort.
4. Generic programming with void* (Quick sort using void *)
5. Variable list of arguments.
6. volatile memory (used in ISR) and (pointer *)NULL) -> offsetof

3) Explore Deterministic Finite State Machine | Flowchart | Storage Classes— Chapter 18. Self-contained Structure Abstraction, such as with array of pointers to function, nested structure.

4) Basics in Abstract Data Type – Chapter 19

a. Linked List— single | double | circular . (Explore simple tree structure)

b. Basic Stack (LIFO)— Push / Pop

c. Queue (FIFO) concepts— Enqueue / Dequeue (with/without own memory pool)

5) System Signal (interrupt) - Chapter 24

6) Quicksort (Divide and Conquer Algorithm). (Implementation)

7) Minimax Tree with DFS : (Class Note).

a. Minimax and Alpha Beta Pruning (DFS | Tree Structure | Basic Backtracking Algorithm)

8) Backtracking Maze algorithm with BFS : (Class note)

9) Abstraction Layered programming

To conclude Level III:

- **This level will conclude the usage of the Dr. K.N. King's Book.** Students should skim thru Ch. 21, 23 to 25-27 on their own, as they all should be used for reference and self-study purpose only.
- May Practice on several USACO Silver and/or Gold level exercises depending on the timeframe and students' analytical skills.

Level IV

Non-Linear Data Structure Algorithms—1 (with C)

There are two parts for non-linear Data Structure. Level I utilizes mostly C. Both focus on learning through pragmatic application along more the vernacular necessary in data structures used in some more commonly used algorithms.

You will implement more complex non-linear data structure instead of just using pre-built APIs.

As always, our style is learn through application. For example, part of non-linear data structure is Heap tree which is widely used in many industrial algorithms. Instead of drilling into , for example, an AVL tree structure, we start with solving a problem / algorithm requires AVL tree implementation. This enhances the interests level and allows students to see the application upfront.

Learning Outcome:

Students will have covered some of the most commonly discussed algorithms / techniques used in many problems which require performance.

Students who are able to complete this level will have proven themselves excellence in not just programming, but also in computational problem-solving . All of the students who can reach this level should gain a highly competitive edge in high school paid internship, as well as college internship engineering/software development programs.

They will also bring them more familiar with the vernacular in algorithmic reasoning and analysis, and implementation, including a variety of techniques in algorithm design. Be able to analyze the Big O running time of an algorithm or method

Learning Resources :

1) **Mastering Algorithms with C:** Useful Techniques from Sorting to Encryption. ISBN -13: 978-1565924536, or ISBN-10: 1565924533. This book will be used for Level IV with additional notes.
2) Class Notes and online packets.

LEVEL IV

Non-Linear Data Structure Algorithms—1 (with C)

1) About running time Complexity for algorithms— Ch.4
    a. Big-O
    b. NP-and Non-NP Complete
2) Discussion in Complexity with previous work/projects:
    a. Introduction to Data Structures and Algorithms —Ch. 1
    b. Advanced Pointer Manipulation — Ch. 2
    c. Recursion—Ch.3
    d. Linked List—Ch.5
    e. Quick Sort vs Heap Sort vs Merge Sort—Ch.12
    f. Binary Search—Ch.12
3) Event Handling—Queue (Ch. 6)
    a. Queue / Dequeue pointer to functions
    b. Explore background processes
3) Hashing—Ch.8 Hashing
    – Linear Probing,
    – Open addressing
    – Collision avoidance
    – Creating your own data dictionary table—insert, lookup, remove
4) Math Expression Processing—Stack—Ch. 9
5) Huffman Coding (Loss-less compression)
    a) Self-balance BinaryTree (AVL)
    b) Priority Queue | Heaps (Tree) — Ch. 10
    c) Greedy Algorithm.—Gr. 14
6) Dijkstra Algorithms (Graph) —Ch. 7 & 11
7) Numeric Methods—Ch.13
    – Foundation of Regression Analysis—Polynomial interpolation and Least Squares Estimation
8) Geometric Algorithms with Convex Hull —Ch. 17

Note: Ch 15 and 16 will be covered in Level V with C++ instead.

Level V

Non-Linear Data Structure with Algorithms—2 (C++ & STL)

This course will switch you completely into C++. You will continue to gain deeper understanding in the techniques and data structure implementation skills required by more complex system software and algorithms design. In addition, your work will achieve higher productivity.

Learning Outcome:

By the time this level is completed, students should have a sound grasp of Object-Oriented **Design Pattern**. If you are going into computer engineering or any courseware which involves working with micro-controller libraries, completion of this will help you greatly in mastering utilization of these libraries, as well as design aspect.

Students will gain the advantage of using Standard Library (STL) Abstractions to reduce complexity of their own program solutions, increase productivity, but still understanding the C++'s extra features coming with overhead. Besides, C++ exemplifies the usage of abstract data structure with reduced amount of low level detailed work as well.

This is designed for students who are interested in Robotics Engineering and AI learning; both realms require high performance. Most resources / libraries written in high level languages for embedded systems are in either C++ or C .

C++ is a superset of C, it means it will contain many important features which will reduce your work load vs using just C, especially in arrays, list, string, memory management, and non-linear data structure implementation.

(Do note: highly recommend to go into his level if you wish to continue working on USACO Gold+ Level. STL will alleviate much complexity in your coding.)

However, nothing can replace with the passion of "problem solving" and "Practice!"

Books and Learning Resources:

- Data Structures and Algorithm Analysis in C++ : Edition 2013 . ISBN-13: 978-0273769385, or ISBN-10: 0273769383
- Class notes (a lot of it)

Covered Topics

C++ Language Features (differences from C)

1) C++ variables and functions (including reference and overloading)
2) Intro to streams: cout and cin vs. C standard File I/O
3) Namespaces vs. header files
4) New and delete memory allocation
5) Auto and decltype

Objects and Classes

(A) The Basics

1) Members and methods
2) Member access control
3) Constructors and Destructors
4) Class scope
5) Const and static members
6) Comparison to structs
7) The *This* pointer
8) Friend functions
9) Operator overloading (including C++20 <=> operator)

(B) Inheritance

1) Basics
2) Polymorphism
3) Abstract Base Classes
4) Inheritance and dynamic memory allocation:
5) Multiple inheritance

(C) Template classes

1) Generic programming

2) Template class
3) Auto -> decltype() return
4) Friend functions and Template specialization

Additional language Features

1) Move semantics and Rvalue references
2) Exceptions
3) Runtime Type Identification and Casting (const_cast and reinterpret_cast)
4) Lambda Functions

The Standard Template Library

1) String Class
2) Smart Pointers – in depth
    What is RAII (Resource Acquisition is Initialization)
    1 Scope-Bound Resource Management
    2 Grabbing and releasing resources for memory, sockets, etc.
    3 How do Smart Pointers apply to RAII
3) STL Containers (greatly enhance productivity vs pure –C)
    a) Sequence Containers - vector, linked list
    b) Container Adaptors - Queue, Priority Queue, Stack
    c) Associative Containers
        – Ordered Set, Multi-set, Map, Multi-map
        – UnOrdered Set, Multi-set, Map, Multi-map
    d) Range based For loops
4) Iterators
5) Algorithms
6) Functors
7) Initializer_list

Input, Output and File IO

– iostream, File streams and Stringstream

Algorithms Checklist

With C

1) MinMax & Alpha-Beta Pruning
2) Maze Navigation (Breath First Search / Queuing Structure)
3) Huffman Coding Compression
4) Math Expression Parsing—infx to postfix
5) Longest Common Subsequence
6) Longest Common Substring
7) Longest Common Subsequence
8) Longest Common Increasing Subsequence
9) KNN
10) FloodFill

With C++

1) Dijkstra's Shortest path
2) Johnson Algorithm
3) Kruskal's Minimal Spanning Tree
4) Prim's Minimal Spanning Tree
5) Travelling Salesman Problem
6) A* algorithm
7) Classroom scheduling
8) Knapsack Algorithm with Dynamic Programming as well
9) Lempel Ziv (LZ78)
10) SHA-1 or 5
11) Genetic Algorithm (for job scheduling)

Common Techniques

- Generic Programming
- Dynamic Programming
- Greedy Method

Others (Tentative):

- ☐ Parallel programming with semaphore.
- ☐ Socket Programming

Level V

Storming Robots 3322 Rt. 22 West, Building 15, #1503
Branchburg, NJ 08876
Phone: 908-595-1010 Fax: 855-595-1010
office@stormingrobots.com