CONTENTS

ABOUT THIS TRACK

Computer Science (CS) skills should go far beyond just programming and syntax. It should focus on problem-solving ability with computational thinking (CT), even for grade schools. CS with CT skills is indispensable to prepare our generation for the world of artificial intelligence.

Storming Robots utilizes Robotics to animate problem-solving efforts starting in Grade 5. However, we encourage students to study in this Algorithms in C/C++ Track starting no later than Grade 8. This syllabus consists of five levels. The levels aim to bolster students' technical ability to take on technical projects which require them to create proficiency software.

Characteristics (important to read first)

1) Go far beyond just syntax, but algorithmic thinking (computational thinking) along the way, and how to build high-quality software. At the later levels, students will engage in more sophisticated algorithms.
2) **Focus on problem-solving**/software development skills, so students will not work with the physical robot.
3) **Heavily on analytical skills development, NOT by rote memorization**.
4) **Progress is self-paced**. All assigned exercises should be completed with excellent quality. In the later levels, more emphasis on robustness, memory consumption, readability, maintainability, etc. All levels stress in Computational Thinking and Efficiency.
5) Students will be allowed to move quickly to the next concept after demonstrating satisfactory understanding through their homework and/or on a pop-quiz in class if necessary. Those with gaps in their prerequisite knowledge will receive additional exercises to address the shortcomings.
6) Most exercises/instructions constantly require higher order of thinking. (i.e., no spoon-feeding method).

Levels Aligning with Storming Robots Roadmap

Allow students to embed competitions and standardized exams in-between levels. Therefore, most of the homework assignments provided in the later levels are not from the book but projects adhering to professional quality. The following shows a general summarized idea of expectations.

1) Students who completed level I with high "proficiency" have the capacity to self-study for Advanced Placement Computer Science (AP CS A). The majority of these students score 5 in AP with a small extent of review on the fundamental object-oriented structure.
2) Some Level II students participate in the USA Computing Olympiad (USACO) online exam, and ACSL. Do note that both exams can be very challenging even to the best students, and far more demanding in analytical and computational programming skills than AP CS A. One of the common traits among the students with high proficiency in upper levels is their ability to adapt to other technologies.
3) Level III students participate in most of SR's advanced robotics activities. High proficiency in this level matches the professional quality, and gain a higher competitive edge to Internship available for high school students in the USA – see https://mp.stormingrobots.com.

How to do Well in this Track?

Best Practices:

1) Review the book materials:

   — Even after concepts are explained in class, revisit the textbook.

   — Jot down questions and focus on the Q&A sections for deeper insights.

   — Ask questions: Bring questions to class for discussion.

2) Complete Exercises Thoughtfully:

   — 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 acquire the analysis process to arrive at the solutions. Do not allow yourself to be satisfied just because a program set seems to be complete, but only after you feel you do understand.

   — Test Extensively— Don't limit your testing to provided samples; think "what-ifs" and broaden your test inputs.

3) Indispensable process of analysis:

   — The cornerstone of true learning is to go through the process of analysis independently. It's important to engage deeply with a problem by breaking it down, experimenting, and learning from mistakes; NOT merely implementing a given solution.

   — Limit acquiring external assistance without putting effort in trying to solve problems. Receiving substantial external assistance at home to complete assigned work often will quickly cause you to miss out on the critical process of analysis. This indispensable learning element strengthens your ability to apply in various scenarios.

   — Stay perseverant in taking on challenging problems. Ensure that you first try to work through a problem on your own, even if it means struggling a bit, before seeking outside help.

4) Avoid Rote Memorization:

   — Simply memorizing a technique without understanding the underlying logic will leave you unprepared when the problem is presented in a new or slightly tweaked form. Strive to understand the 'why' and 'how' behind each method.

5) Practice, Reflect, and Iterate:

- This is a journey—complete with mistakes and adjustments—is essential to developing strong analytical skills that you can apply in any future more advanced activities such as exams, competitions, and projects.

6) Dedicate time and focus:

- Should expect approx. 3+ hours of programming homework per week, with at least an hour each time for homework.

7) Utilize the Debugging tool :

- Not only helps you quickly locate and resolve issues, but also foster deep engagement with your code.
- Actively use these tools in a proper way, you gain in-depth understanding of the program's inner workings, which reinforces learning and sharpens your problem-solving skills

Danger of Receiving too much External Assistance

Relying on external help before fully engaging in your own analytical process can hinder your development in critical ways:

1) Lack of Self-Reliance: Early dependence prevents you from cultivating the confidence to solve problems on your own.
2) Shallow Engagement: Without digging deep into the problem, you miss out on understanding the underlying principles.
3) Lack of Critical Thinking: Simply implementing given techniques, without analysis, means you won't develop the ability to think critically on your own.
4) Limited Creativity, but just being passive: Over-reliance makes you a mere do-er instead of someone who can innovate and adapt when problems change.

Bottom line: The indispensable process of analysis is the cornerstone of true learning. True mastery comes from working through problems independently—making mistakes, learning from them, and developing your own problem-solving strategies. While help from sources like Stack Overflow or family members can be beneficial, it should only complement your journey, not replace it.

Requirement for Level Promotion:

1) MUST demonstrate a satisfactory level of understanding thru their work – see "What was deemed to be complete with an assignment?"
2) What was deemed to be "complete" with an assignment?

- Students must be able to explain their work, not just the syntax but thought processes. If a student fails to explain their work, instructor will require the student to take a mini-test with similar but tweeted problems.

- 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.
- All completed "well-tested" and approved source codes must be uploaded onto the student work folder.
- MUST upload your work onto your student google folder (provided to you from Storming Robots).

3) MUST pass pop-quizzes.

****** DO NOTE: Completing all exercises naturally follows after deep learning, but not necessarily vice versa.**

Learning Resources—Books and online packets:

Level B to II :

- C Programming: A Modern Approach, 2nd Edition by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503.
- Additional Supplemental (optional) and Storming Robots online learning site — Computer Science Track.

Level III :

- Part of K.N. King book
- Class Notes

Level IV:

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

Level V:

- C++ Primer Plus (Developer's Library) 6th Edition, by Stephen Prata. ISBN-13: 9780321776402.
- (Optional) : Data Structures and Algorithm Analysis in C++, 4th Edition by Mark A. Weiss—ISBN-10: 0273769383
- Class Notes.

Software required

For Windows OS: Microsoft Visual Studio Community version.

For Apple OS, you will may use one of the followings :

- an online tool, such as online-GDB IDE.
- Microsoft Visual Studio Code version.

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:

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

Violation will disqualify you from participating in any competitions with Storming Robots.

Should you aim for all levels?

No, it depends on your interests and strength.

Level I: Anyone who wishes to study any engineering and science major; minimal requirement for entering our Robotics and Electronic.

Level II & III: Gain entrance to more advanced robotics activities thru SR. Most students who wish to engage in more advanced robotics that require more sophisticated algorithms will take II/III simultaneously with Robotics at SR.

Level IV & V: Particularly for future Computer Science majors in college or anything requires heavy software development skills.

⏮_Level_B-I_: I II III IV V

LEVEL B-I : THE FUNDAMENTALS

Learning Outcome

1) Students, who accomplish this level with high proficiency, will find it very easy to self-study Advanced Placement in Computer Science and achieve a high score. A large majority of them reported to us that they all achieved a score of 5 with ease.
2) By the time this level is completed, students should have completed approximately 40 programming exercises up to Chapter 8 from the K.N. King book and others small ones under struct and enum.
3) Possess knowledge in programming fundamentals with an emphasis on producing clear, robust, and reasonably efficient code using top-down design, followed by informal analysis and effective testing and debugging.
4) Build the habit in analytical thinking, NOT just syntax recall.
5) Build the mindset with efficiency in mind.
6) Experience the importance of proper testing
7) Experience the importance of using Debugger for troubleshooting.
8) Gain experience in using one of the most versatile integrated development environment
9) Good understanding in utilizing Linear data structure – array, fundaments in stack, queue.
10) Know how to create simple reusable functions with arguments.
11) For Level B (up to ch.6): students should be able to do simple control structure, take a very straightforward problem set to resolve it with computer programming in C.
12) For Level I (up to ch.8 + basic enum and struct), students should master nested control structure, analyze a problem set, and figure out how to resolve it with computer programming in C.

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

Learning Resources:

1) Book required: C Programming: A Modern Approach, 2nd Edition by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503.
2) Class Notes and online packets.

Covered Concepts:

Level B

1) How to use the Microsoft Visual C/C++ Compiler IDE.
2) C Fundamentals in writing Simple Program and Standard Formatted I/O - Chapter 2, 3

3) Fundamental Expressions - Chapter 4
4) Selection/ if - Control Statements | Boolean Expressions - Chapter 5
   - How the selective "if" interprets a complex expression
   - Understanding Boolean expression vs numeric expression .
5) Loops Control Structure - Chapter 6
   - Short-circuit evaluation of AND & OR.
6) Rudimentary level understanding in creating functions : - (partial Ch.9 & 10 with Class Note)
   - Primitive return data types.
   - Simple primitive input arguments/parameters.
7) Fundamental understanding in using Multiple files within a project - (partial Ch.9 & 10 with Class Note)
8) Base Conversions—Hexadecimal, Binary. (with Class note)
   - Understand why hex is important.

Level I

1) Primitive Data Types
   - Signed vs unsigned.
   - Understand the data range
2) Learn how to use the debugger – Watch / conditional break point
3) Chapter 7 (plus Class note)
   - Continue to polish their understand in creating nested selective structure
   - Reinforce readability, maintainability, effectiveness of nested loops, robustness
4) Linear Data Structure: 1D and 2D Array – Chapter 8 + class notes
   - Understanding more in-depth regarding its memory layout
   - How that represent matrix/tables
   - Explore what is an adjacency matrix (fixed size)
5) Abstract Data Type: Enum – Fundamentals (with Class note and Partial ch.16)
6) Abstract Data Type: Struct – Fundamentals with Class note and Partial ch.16.
   - Stress in encapsulation and reusability

7) Abstract Data Type: Class - Rudimentary level of Object-oriented Programming Structure

   - Anatomy of a Class structure
   - Overloading / Overriding
   - Private vs Public
   - Static vs instance

8) How to create a project app using a Class and its header file.

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 I:

Students must demonstrate satisfactory ability in both mechanics and analysis portion of all content listed above.

What's next:

***** MUST READ: Advanced Placement in Computer Science A:**

***** Major change in AP CS A :**

A Starting in 2025-26 school, College board may decide to remove inheritance. However, as always, at SR, our focus remains on building strong computational and critical thinking skills - not just the mechanics of the language. The key strength of Object-Oriented Programming (OOP) design (that's what Java's basic fundamental framework) lies in its ability to:

- model real-world entities and their interactions through "objects,"
- promoting modularity, reusability, maintainability, and understandability of complex problems by encapsulating data and behavior within self-contained units,
- making complex systems easier to manage and adapt to changing requirement

Thus, our *Algorithms in C/C++ Level II* will still include a basic introduction to inheritance. We believe this foundational concept enriches students' understanding of software design and equips them with adaptable skills for evolving industry needs, not just about taking the AP exam.

How does Level I align with the AP CS- A?

At the completion of level I, the only topics in AP CS-A not yet covered are:

- ArraysList (covered in level II)

- Simple inheritance and polymorphism (covered in level II)
- Polymorphism are outside the scope of the AP Computer Science A course and exa potentially will be excluded starting in 2026 AP season.)
- Recursion. (covered in Level II)
- Beyond AP CS A - Designing and implementing inheritance and polymorphism relationships.

A large majority of our CS students reported to us that they all achieved a score of 5 with great ease. You are recommended to read up on fundamentals in object-oriented principles in design pattern. (Using the AP Barron AP CS A book will be good.) Or, take an AP CS at SR during the summer – SR's AP CS classes will also cover more data structure and problems analysis beyond the bare mechanics covered in AP CS – A.

Recommendation after achieving proficiency in Level I?

- **Interested in High School Robotics Electives, i.e. competitions (either Hardware platform or Simulation) ?** Must possess proficiency by passing our Level I Test. May view the programs and competitions criteria online.
- **Continue on Algorithms in C/C++ - Level II.** Many do both Algorithms in C/C++ along with Robotics with Electronics.

⏮_Level_B-I_: I II III IV V

LEVEL II : DATA ABSTRACTION TYPE (ADT) - LINEAR STRUCTURE

This level involves software development skill well beyond Advanced Placement CS-A. Students should expect to work lesser number of exercises, but more demanding than Level I.

Learning Outcome:

1) Should be able to tackle American Computer Science League Junior to Intermediate Level.
2) Should be able to tackle Bronze/Silver in the USACO. Based on our history, all students who demonstrate high proficiency in completing Level II and possess a high chance to be promoted to the Silver Level. Some even got Gold – see our students' achievements here. Do note that the target is not just about participating in USACO. Achieving Silver in USACO is just a by-product of excellent preparation and building a dynamic knowledge base to prepare our students to participate in more challenging opportunities, such as competitions, summer engineering, and research internships, etc.
3) Strengthened programming analysis. Most programs exercise you have completed in even Level II under this program require higher analytical skills - FAR BEYOND what is required in AP Comp. Sci. A.
4) Heavily focus on robust implementation and error checking
5) Become more efficient in Debugger, including breakpoints, observation of variables, watch feature.
6) Building stronger mindset of computational thinking.
    - Ability to recognize patterns, pay attention to reduction for efficiency, and algorithmic thinking instead of brute-force.
7) Should be able to conduct a more complex project than typical College upper-year level in CS.
8) Gain a better understanding of the compilation and link process and the complexity of building larger projects consisting of multiple folders and files.
9) (tentative) Get experience in versioning control system—Git.

Learning Resources:

- Book required: Same book for previous level. C Programming: A Modern Approach, 2nd Edition by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503.
- Class Notes and online packets.

Covered Concepts:

1) Review of the number base system, solid understanding on data types and their ranges, etc... (MUST pass the LEVEL-II-beginner Test)
2) More advanced analytical work on Arrays (class notes)
    - More focus on Multi-Dimensional

3) Recursions – from simple to complex (with class note)
   - include very basics in dynamic programming method.
   - Binary recursive: More seasoned in recursive programming style.

4) Memory pointers vs. 1D and 2D array – Chapter-11 & 12

5) Explore another Linear Data Structure: Stack and Queue, and Priority Queue. (class notes)
   - Conceptual only.
   - Implementation in Level III

6) Error Handling – Ch.24.1 to 24.2. and class notes with "try … exceptions.

7) String manipulation:
   - C-String: Chapter-13
   - C++ string manipulation - Quick transition from C-String to C++-String.
   - Learn how to use the C++ vector for string

8) Dynamic allocation — Chapter 17-1 to 17-4 - malloc/free, calloc/free, vs. realloc . Utilize memxxx() functions

9) Command-line arguments

10) Preprocessor - Chapter-14 —Conditional Compilation, such as #if, #elif, #line, ##, etc. Its role in compilation process.

11) Bitwise operations — Chapter 20

12) Commonly used Search / Sort algorithms (both in iterative and recursive) :
   - Binary Search
   - Insertion sort (implementation)
   - Qsort with integers list (using the API)

13) Basic File I/O — Chapter 22 (with C & C++ - add'l note)

For American Computer Science League Competition Students only:

- Boolean Algebra

- LISP
- LOGIC Gate
- Explore Prefix/infix/postfix (for solving complex Math Expression) - conceptual ONLY. (class notes)

To conclude Level II:

- This level involves more mini-projects and exercises external to the book.
- Summary of the chapters covered (not necessary in the numeric order): Chapter 9 to 16, and 20, 22.

What's next:

- Should continue working on the USACO online practices. See Program criteria for more suggestions.
- Continue on Algorithms in C/C++ - III.
- And/or participate in more advanced robotics competition.

⏮_Level_B-I_: I II III IV V

LEVEL III : DATA ABSTRACTION TYPE (ADT) – NON-LINEAR STRUCTURE

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

The 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) Should be able to tackle Silver/Gold in the USACO. Based on our history, all students who demonstrate high proficiency in completing Level III possess a higher chance to be promoted to Silver or Gold level.
2) Complete College level Data Structure.
3) Explore Inheritance and polymorphism design.
4) Improvement in modular programming design, including abstraction layers. You will create your own libraries for applications.
5) Equipped with sound programming skills surpass most undergrad CS capacity. Now, you are truly stepping into Computer Science — as opposed to programming — such as abstraction, correctness, complexity, and modularity.
6) Gain competitive edge in HS Summer internship. Gain opportunities at Storming Robots' advanced competitions
7) Higher ability in solving complex projects and Gain exposure how simple assembler instruction sets work.
8) Be able to operate closer to the machine's inner workings, such as memory, processing power, with less abstraction, such as exploring assembly language structure.
9) Should feel comfortable to navigate around Linux System, and be able to build programs and libraries yourself with gcc and g++ and makefile.
10) During November and April, students are encouraged to continue competing in ACSL – intermediate / Senior level and or working on USACO problems set and take the Bronze to Silver Level online Exam.
11) (Optional) Basics in using Versioning Control System—Git.

Higher Expectation in the following:

- Error Checking
- Well tested

- Concise and elegant , and well-documented
- No program memory leaks

Learning Resources:

1) *Book required:* Same book for previous level. C Programming: A Modern Approach, 2nd Edition by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503.
2) Class Notes and online packets.

Covered Concepts:

1) Try out a few USACO problems and submission.
2) System Signal (interrupt) - Chapter 24.3 & 24.4
3) Advanced understanding in Advanced memory pointers.
   - pointers to functions, pointer to struct
   - some fundamentals about memory – such as data alignment, padding, big vs little endian
   - Advanced Uses of Pointers —Chapter-17.7 and Chapter-18, and class notes such as generic programming with void* with Quicksort (Divide and Conquer Algorithm). (Implementation)
4) Advanced Struct, Union, Enumeration, Bits-structure — Chapter-16 and class notes . Such as
5) (optional) Advanced File I/O – memory buffering , redirection
6) Implementing the Last of Linear Data Structure: Linked List— single | double | circular. Adjacency list.
7) Implementing Non-linear data structure
   - Create their own stack / queue library: Basic Stack (LIFO)— Push / Pop and Queue (FIFO) concepts— Enqueue / Dequeue (with/without own memory pool)
8) Solving Math Expression - with precedency order
   - Complex expression from infix to postfix (class notes)
9) Implementing Simple Binary tree.
10) Implementing simple Graph structure - build Adjacency list to represent a graph structure
11) Binary-index Tree (Fenwick)
    - efficiently calculating prefix sums and performing range updates in an array

For American Computer Science League Competition Students only:

- Explore FSAs and Regular Expressions
- Introduction to Digital Logic Gate.
- Explore how to read a simple Assembler Language.

To conclude Level III:

- This level will conclude the usage of Dr. K.N. King's Book. Students should skim thru Chapter 21, 23 to 25-27 on their own, as they all should be used for reference and self-study purposes only.
- Will complete a final larger scale real-world application project that conglomerates all concepts learned thus far.
- May Practice several USACO Silver and/or Gold level exercises depending on the timeframe and students' analytical skills.

⏮_Level_B-I_: I II III IV V

LEVEL IV - NON-LINEAR DATA STRUCTURE AND ALGORITHMS I (WITH C)

There are two parts of non-linear Data Structure. Part I utilizes primarily C. Both focus on learning through practical application along with more the vernacular necessary in data structures used in some more commonly used algorithms.

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

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

Learning Outcome:

1) Students with high proficiency in software development, including building "Abstraction Layers"
2) Should be able to tackle Gold in the USACO.
3) Become more familiar with the vernacular in algorithmic reasoning, analysis, and implementation, including various techniques in algorithm design. Be able to analyze the Big O running time of an algorithm or method
4) May be selected to take on an instructor role at Storming Robots and possibly other opportunities for leadership skills.
5) Have covered some of the most commonly discussed algorithms/techniques used in many problems which require performance.
6) Gain a highly competitive edge in high school paid internship and college internship engineering/software development programs.
7) Build a foundation for future studying in A.I. This course focuses more heavily on A.I. concepts, including formulating search problems, adversarial games, uninformed searching, and informed searching. Students will use these skills to build
8) Able to pursue more complex algorithms and conduct advanced projects on their own.
9) Has become adept to grasp AI tools on their own. With Linear Algebra knowledge, you will be well equipped to learn other AI algorithms, not the tools.

Learning Resources:

1) *Book required:* 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.

Covered Concepts:

1) Review :

- Introduction to Data Structures and Algorithms —Chapter 1
- Advanced Pointer Manipulation — Chapter 2
- Analysis of Algorithms — Chapter 4
- Trees—Chapter 9
- Stacks and Queue with Event Handling – Chapter 6

2) Explore NP-completeness

10) Graphing algorithms:

- Adversarial Search (DFS)
    - Minimax Tree with Depth First Search (for Zero-sum games) (Class Note).
    - Alpha Beta Pruning (DFS | Tree Structure)
- Backtracking Maze algorithm with Breath First Search : (Class note) : non-linear data structure: Graph structure). Maze Navigation
- This may be done in a maze setting with two possible challenge: Traverse the whole maze to seek for location of a certain target. BFS to go back to the start point to report the location of the targets

11) Heuristic search – A* (class note)

12) Binary Tree Algorithms: (with class notes)

13) Heaps and Priority Queue—Chapter-10

- Loss-less Data Compression with Huffman Coding — Chapter 14 (Greedy Algorithm)
- Binary Search Tree (completed sorted)
- Self-balance Binary Tree (AVL)
- (optional) Red-Black Binary Tree

14) Hashing—Chapter 8

- Chain linked vs Open addressing Methods
- Understanding loading factor and Collision avoidance
- Evaluate between two implementations with larger datasets
- Creating your own dictionary structure and searching with hashing method

15) Sets and Graphs – Chapter 7 & 11

16) Dykstra Algorithm.

17) Minimum Spanning Tree

18) Knapsack Algorithm (including items and using dynamic programming).

19) Numerical Methods—Chapter 13

20) Genetic Algorithms

21) Geometric Algorithms with Convex Hull —Chapter 17

To Conclude Level IV

— Completion of all given projects given in this level.

⏮_Level_B-I_: I II III IV V

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!"

Learning Resources:

1) *Book:*

   a. C++ Primer Plus (Developer's Library) 6th Edition, by Stephen Prata. ISBN-13: 9780321776402.

   b. (Optional) : Data Structures and Algorithm Analysis in C++, 4th Edition by Mark A. Weiss—ISBN-10: 0273769383

2) Class notes (a lot of it)

Outcome

Expect to reach professional level of software development knowledge. Students will gain highly competitive edge in seeking internship which requires software development skills.

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
6) vector, list, deque, set, multiset, map class

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)
   - Scope-Bound Resource Management
   - Grabbing and releasing resources for memory, sockets, etc.
   - How do Smart Pointers apply to RAII
3) STL Containers (greatly enhance productivity vs pure –C)
   - Sequence Containers - vector, linked list
   - Container Adaptors - Queue, Priority Queue, Stack
   - Associative Containers
4) Ordered Set, Multi-set, Map, Multi-map
5) UnOrdered Set, Multi-set, Map, Multi-map
   - Range based For loops
6) Iterators
7) Algorithms

8) Functors

9) Initializer_list

Input, Output and File IO

- iostream, File streams and Stringstream

Advanced Topics (Optional)

1) Multi-threading
2) Socket Communication
3) More advanced performance features in C++17 & up

To Conclude Level V

Revise several algorithms, such as

1) Huffman Coding
2) Shortest Path with Dijkstra
3) Shortest Path with Minimum Spanning Tree
4) KnapSack
5) ML Algorithms such as Linear Regression, KNN, K-means And more (depending on progress)

ALGORITHMS CHECKLIST

This list is a sample list, and may be modified. Students will have completed their own implementation, not using intrinsic algorithms api or data structure. Yes, students write their own implementation.

Greedy algorithms are faster and use less memory, but they may not always find the best solution. Dynamic programming can find the best solution, but it can be more complex.

With C

Divide and Conquer

1) Binary Search, Quick sort
2) Maximum subarray
3) Quicksort (Divide and Conquer Algorithm). (Implementation)
4) Fast fourier transform
5) Convex Hull
6) A*
7) Shortest 2 points distance
8) BFS – Maze algorithm

Greedy Algorithm Technique:

1) Dijkstra's algorithm (the shortest path from a single source node to all other nodes)
2) Floyd-Warshall (for the shortest paths between every pair of nodes)
3) Bellman-Ford (similar to Dijkstra's algorithm, but handles also negative weight)
4) Scheduling
5) Huffman Compression

Dynamic Programming Technique:

1) Fibonacci
2) 0/1 Knapsack
3) Longest Common Substrings

4) Matrix Chain Multiplication

With C++

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

COPYRIGHT

Unless otherwise noted, Storming Robots retains with copyrights under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0; pursuant from the day this document was published by Storming Robots. You may copy, distribute and transmit the work IF AND ONLY IF all of the following criteria are met.

- Give appropriate credit to the source
- Provide a link to the license
- changes were made only under Storming Robots' permission
- Retain the identification of the creator(s)
- Retain a copyright notice

Please see the Legal Code under the Creative Commons.