CS Exit Ethiopia

Complete Study Notes

Computer Science Exit Examination

1 selected course | 18 November 2026

Selected courses

01. Computer Programming

https://csexitethiopia.com

abelasnake2003@gmail.com | Telegram: @Abechu7

Course contents

Each course begins on a new page. PDF bookmarks provide question, study-note section and revision-topic navigation.

Computer Programming 6

How to use these study notes

START01 A practical study cycle

Start with a small set of questions before reading.

Mark the exact thing that stopped you: an unknown term, a rule, a calculation, a trace or a choice between alternatives.

Use the question lookup to open the relevant section.

Read one section, close it, and explain the rule in your own words.

Solve the question again and say why each other choice fails.

Then change one assumption or input and solve the variation.

This checks understanding beyond recognition.

A useful 60-minute session: 10 minutes of cold questions, 20 minutes of focused reading, 20 minutes of closed-book solving and 10 minutes updating an error log.

Adjust the timing to your needs.

Revisit mistakes after a gap, for example the next day, several days later and the following week.

These intervals are a practical starting plan, not a fixed learning law.

Record the section ID, your mistaken rule, the corrected rule and one boundary case.

Keep the log short enough to review.

Mix related courses during revision: programming with data structures; databases with security; networking with administration.

Explain how the same word changes meaning across contexts.

START02 What to emphasize

Emphasize definitions, key differences, algorithm costs, SDLC models, databases, networking and security.

Use short oral questions to check understanding and correct common misconceptions.

Combine those habits with the tracing, calculation, design and diagnosis demanded by the Ministry learning outcomes.

Allocate revision by both exam share and your error log.

A course with fewer items can still deserve more time if your foundations are weak.

The blueprint is a planning baseline, not a prediction of exact questions.

Definitions: distinguish nearby concepts, not just their names.

Calculations: write units, formulas, assumptions and intermediate steps.

Code and SQL: check validity before tracing the text; state any changes before analyzing a corrected version.

Algorithms and protocols: know the input, decision rule, state change, termination and failure cases.

Design choices: state the criterion.

A choice can change when cost, reliability, consistency or latency matters more.

Course group	Items in the 100-item blueprint	Revision implication
Web Programming	9	Reserve extra practice for HTML, CSS, JavaScript, PHP, client/server behavior and safe data handling.
Data Structures and Algorithms	7	Trace operations and compare their assumptions and costs.
Each of the other fourteen courses	6 each	Study all sixteen; do not abandon a whole course based on a short note or an alleged repeated-question list.

START03 How to handle exam traps

Underline NOT, FALSE, BEST and EXCEPT before comparing choices.

Check whether the stem supplies enough facts to select one answer.

A long scenario is not automatically an analysis question, and a familiar answer letter is not evidence.

When code is invalid, a diagram is absent, or two choices express the same rule, identify the clarification needed before answering.

Learn the correct rule and the smallest clarification needed.

Do not memorize an arbitrary key as a universal fact.

Distinguish worst case from average case, and an upper bound from a tight bound.

Check language version, SQL dialect, addressing assumptions, graph weights and tie-breaking rules.

For subnetting and scheduling, calculate rather than guess from a familiar number.

Use timed mixed papers only after you can explain your errors.

Revisit the reading sections behind recurring mistakes.

Computer Programming

Complete Study Notes

CP01 | Turn a problem into a program

Identify inputs, outputs, legal ranges and edge cases before coding.

Solve one small case manually, choose an algorithm, express it as pseudocode or a flowchart, implement, test and maintain.

For the largest of three numbers, initialize best to the first input, compare the second and third, then report best.

Initializing best to zero fails when all inputs are negative.

Programming includes writing, testing, debugging and maintaining instructions.

A program can execute without solving its intended problem; compare its actual result with the required result.

Computer Programming module

CP02 | Algorithm properties and paradigms

An algorithm gives precise, executable steps and terminates for every legal input.

Definiteness means each step has an unambiguous meaning; effectiveness means a step can actually be performed.

Inputs may be absent, but a useful algorithm specifies a result.

Correctness means the result meets the specification.

Efficiency is desirable and measurable, but an inefficient terminating algorithm is still an algorithm.

Efficiency is also a desirable property of an algorithm.

Procedural programming organizes operations in procedures; structured programming combines sequence, selection and iteration; object oriented programming organizes state and behavior in objects; functional programming emphasizes function evaluation.

A conditional statement is a construct, not a separate programming paradigm.

Computer Programming module

CP03 | Language levels, generations and portability

Machine instructions are conventionally first generation; symbolic assembly is second; general high-level languages such as C++ and Java are third.

Database query and application generators such as SQL are conventionally fourth generation.

Fifth generation is a loose historical category associated with declarative logic, constraints and AI/expert systems.

Generations do not determine whether an implementation uses an interpreter.

Machine and assembly code depend on an instruction set.

High-level source is easier to read and often portable after recompilation, provided it avoids platform assumptions.

People can write both assembly and high-level programs.

Python, Perl and PHP are commonly called scripting languages; C++ is conventionally compiled.

These labels describe common use, not an absolute technical prohibition.

Computer Programming module

University of Rhode Island: language levels and generations

CP04 | Compilation, linking, loading and Java translation

In the conventional C++ toolchain: source → preprocessing → compilation proper → assembly → object files → linking → executable → loading → execution.

Preprocessing handles directives such as #include; compilation may emit assembly; the assembler emits machine instructions in an object file.

The linker resolves external symbols and combines object files and libraries.

The loader places an executable and required components in memory and starts it.

GCC documents the first four stages explicitly.

A compiler need not emit a complete executable or even native machine code.

Java javac produces bytecode, which a JVM interprets or compiles at runtime.

Interpreters execute a program representation; “one source line at a time” is a beginner approximation.

Compilation uses multiple passes, so “whole program in one step” is not literally universal.

Computer Programming module

GCC manual: compilation stages

CP05 | Identifiers, keywords and comments

In the basic ASCII identifier rules, start with a letter or underscore; continue with letters, digits or underscores.

Do not use a keyword.

C++ is case sensitive: exitExam and ExitExam differ; delete and float are keywords, but FLOAT is a legal identifier. variable_1234 is valid; 4thInput, T-3 and names containing spaces are not single identifiers.

Avoid reserved underscore patterns in application code.

Comments use // to the line end or /* ... */ for a block.

They explain intent and assumptions for maintenance; ordinary comments do not execute or automatically improve speed, reliability or portability. main is the conventional entry function name, not a C++ keyword.

Computer Programming module

CP06 | Declarations, initialization, constants and types

Write type name; or type name = value; for ordinary variables: int age = 20;.

Declaration introduces a name and type; initialization gives its initial value; later assignment replaces a value. a=b copies a scalar value, so later changes to b do not change a. A pointer or Java object reference instead copies an address/reference.

Use int for whole numbers, float/double for fractional approximations, char for a character and bool for logical values in C++.

Type sizes depend on the implementation; do not import Java widths into C++. short n, short int n and int short n denote the same signed short type. const int n=5 and int const n=5 are equivalent; an ordinary const int needs initialization. typedef creates an alias, not a distinct new type.

Computer Programming module

CP07 | Literals, conversions and sizeof

Single quotes denote a character literal, such as 'A'; double quotes denote a string literal, such as "A".

A string includes more storage than just its visible characters when a terminator is required. sizeof reports size in bytes, with sizeof(char)==1; the byte width and other sizes are implementation dependent. sizeof(array) gives total array storage before conversion to a pointer.

With int x=5,y=2, x/y is 2.

Assigning that quotient to double gives 2.0, not 2.5. static_cast<double>(x)/y performs floating division and yields 2.5.

Converting an in-range fractional floating value to int truncates toward zero.

Cast before the operation if a fractional result is needed.

Computer Programming module

CP08 | Arithmetic and compound updates

Multiplication, division and remainder have higher precedence than addition and subtraction; equal-precedence arithmetic groups left to right.

Integer division truncates toward zero.

For integers, a == (a/b)*b + a%b when the division is valid. % requires integral operands in C++; (8.0*3)%5 is invalid.

Division by zero and signed integer overflow cannot be given ordinary numeric outputs.

x+=y, x-=y, x*=y, x/=y and x%=y update x using the operation and conversion to its type.

For simple variables, trace them as x=x op y. Starting p=q=n=8, p+=n gives 16 and p/=q gives 2.

Starting x=10, x+=5 then x/=2 gives 7. x=x+x doubles; it does not generally add one.

Computer Programming module

CP09 | Prefix, postfix and evaluation safety

Prefix ++x changes x and yields the new value; postfix x++ yields the previous value and also increments x. Separate the value used in the expression from the stored value after it.

Starting a=8,b=5, a++ + --b uses 8+4, gives 12, and leaves a=9,b=4.

Starting x=1,y=10,z=-3, (x++*y--)+ ++z gives 8 and leaves x=2,y=9,z=-2.

Precedence determines grouping, not a universal evaluation order.

Avoid expressions that modify a variable multiple times without sequencing.

In C++17 and later, chained operator-form stream insertions are sequenced left to right: with x=5, cout<<x++<<" "<<++x prints 5 7.

Before C++17 this expression has undefined behavior.

For the C++17 expression above, 5 6, 5 5, 6 5 and 6 6 are all incorrect outputs.

Computer Programming module

C++ draft: shift sequencing

CP10 | Boolean, relational, bitwise and short-circuit logic

Relational and equality operators test values; && and || combine truth values; ! negates.

In C++, zero converts to false and a nonzero number to true.

Streams display bool as 0 or 1 unless boolalpha is set.

With a=3,b=5, cout<<(a==b) prints 0.

&& evaluates its right operand only if the left is true; || only if the left is false.

Thus false && dangerousCall() skips the call.

Bitwise & operates on bits and does not short circuit.

With x=1,y=0,z=4, (x&y)||++z evaluates ++z, leaving z=5 and the Boolean result 1.

The precedence ladder is grouping/postfix, unary, multiplicative, additive, shifts, relations, equality, bitwise &, ^, |, &&, ||, conditional, assignment, comma.

Computer Programming module

CP11 | Conditional and comma expressions

condition ? yes : no evaluates the condition and exactly one alternative.

The comma operator evaluates its left expression, then its right, and yields the right value.

Do not confuse it with commas separating function arguments.

For a=6,b=8,x=2,y=4, c=(x>y ? (a--,x) : (b--,y)) takes the false branch. b becomes 7, c receives 4 and a remains 6.

Both alternatives are not executed.

A semicolon in place of the internal comma is invalid syntax; a trace using a comma applies only after that syntax change.

Computer Programming module

CP12 | Trace decisions and recognize flowcharts

A standalone if may execute independently of the following if.

An else binds to its nearest unmatched if; an else-if ladder executes the first true branch only.

In nested if(c1){if(c2) S1; else S2;}, S2 requires c1 true and c2 false.

Braces determine grouping.

With ph=15.3, an invalid-range if can print its warning, and a separate classification ladder can also print Alkaline.

They are not mutually exclusive unless linked by else or an early return.

A flowchart decision is a diamond, processing a rectangle, input/output a parallelogram and start/end a terminator.

A pre-test loop checks before its body; a post-test loop checks after.

Identify a flowchart’s loop from its arrows and the placement of its conditions.

Computer Programming module

CP13 | Switch selection and fall-through

A traditional C++ switch selects using an integral or enumeration value. case labels are constant expressions.

Execution starts at the matching label, then continues through subsequent statements until a break, return or the end. default handles an unmatched value but need not be the last physical label.

If case 2 prints Bus. and case 3 prints Truck. before the first break, input 2 prints Bus.Truck.

A break exits the nearest switch or loop; it does not by itself terminate the entire program.

A case label is an entry point, not an automatically isolated if branch.

Computer Programming module

CP14 | While, do-while and for mechanics

while(condition) checks before each body execution and may run zero times. do {body;} while(condition); runs the body once before checking; it repeats while the condition remains true.

Use it when a menu or input attempt must occur at least once.

for(initialization; condition; update) executes initialization once, then condition → body → update → condition.

A missing condition in for(;;) acts as true.

A for loop is convenient for a known count but can also be condition controlled.

To print indices 0 through n-1, use i=0; i<n; ++i. Starting i=0 with i<5 and i+=2 prints 0 2 4.

Computer Programming module

CP15 | Break, continue, return and goto

break exits the nearest enclosing loop or switch. continue skips the rest of the current loop iteration: a for loop then runs its update; a while loop goes directly to its condition. return leaves the current function and may supply a result.

In main it ends the program normally.

A goto transfers to a label within its function; follow the label reached rather than reading lines in source order.

In the sequence a: print Graduating; goto b; c: print Year; goto a; b: print fourth;, entry at a prints Graduating fourth and reaches the end.

Year is not printed.

Structured alternatives usually make such logic easier to reason about.

Computer Programming module

CP16 | Count iterations and diagnose nontermination

Make a table with condition inputs, body output and updated values.

For i=0,j=8; i<j; i++,j--, body entries are (0,8),(1,7),(2,6),(3,5).

If odd i triggers continue, printing occurs twice.

With outer i=1; i<5; i+=5, the outer loop runs once; an inner j=10; j>i; --j prints nine times.

For n=1; n<=18; n+=2, print n unless n%7==0 then break: output is 1 3 5.

A while(a>b) with fixed a=6,b=5 never changes its true condition.

In data[10]={5,10,...,50}, i=0, advancing i only if data[i]%10==0 causes no progress because data[0]=5.

It prints nothing and loops indefinitely; changing <=10 to <10 alone does not repair progress.

Computer Programming module

CP17 | Functions, parameters and reusable modules

A function groups a named task.

C++ definition: int add(int a,int b){return a+b;}.

Call it with add(x,y);, without parameter type declarations at the call site.

Parameters are names in the definition; arguments are supplied expressions or values.

A void function produces no return value but can have side effects and return; to exit early.

A method is a function associated with a class or object; methods can return values.

Modules group related responsibilities behind interfaces.

Reuse reduces duplication; independent functions are easier to understand and test.

A library provides reusable components; a framework also supplies an organizing structure and often calls application callbacks.

Computer Programming module

CP18 | Value, reference and pointer parameters

A C++ value parameter is a separate object initialized from the argument.

In func(int a,int b){a*=5;b+=5;return a+b;}, calling func(9,3) returns 53 and leaves the caller values 9 and 3 unchanged.

An int& parameter aliases the caller variable, so swap(int& a,int& b) changes both originals.

An int* parameter copies an address; *p=value changes the pointed-to object, but p=anotherAddress only changes the local pointer.

An array parameter int a[] is adjusted to int*: elements can change, and the length must be passed separately.

Java always passes by value, including a copy of an object reference; it does not pass caller variables by reference.

Computer Programming module

CP19 | Overloads, default arguments and inline

Overloads share a name but differ in parameter count or types/order.

Return type alone cannot distinguish them.

Choose a candidate using argument types, then required conversions; report ambiguity if there is no unique best match. int sum(int x,int y,int z=0) allows two or three arguments; defaults occupy trailing positions in ordinary declarations.

inline does not compel the compiler to copy a function body at a call.

Small frequently used functions may be good optimization candidates, but actual expansion is the optimizer's decision.

Inline also has language linkage/definition rules, so it is not simply a speed command.

Computer Programming module

CP20 | Scope, lifetime and persistent static locals

Scope is where a name can be used; lifetime is when its object exists.

A local name is restricted to its block; an outer name may be shadowed.

A namespace/global variable is available where its declaration is visible, not magically in every file.

C++ ::name selects a global name hidden by a local one.

An automatic local is created anew on entry.

A static local is initialized once and retains its value across calls even though its name remains local.

If static x=0 changes by +1 when even and +2 otherwise, five calls return 1,3,5,7,9, whose sum is 25.

Namespace/static storage integers are zero initialized; an ordinary uninitialized local int has no usable assumed zero value.

Computer Programming module

CP21 | Recursion, base cases and call tracing

A recursive function calls itself directly or indirectly.

It needs a base case and progress toward it for all supported inputs. factorial(0)=1 and factorial(n)=n*factorial(n-1) for positive n. factorial(3) expands to 3*2*1*1=6.

Negative inputs require rejection or a separate definition; they do not reach zero under n-1.

Evaluate conditions before expanding a recursive call.

For if(n<m) return 0;, f(1,4) returns immediately; unreachable recursive branches add nothing.

Record each call’s local parameters and pending operation, then unwind returned values.

Recursion commonly uses smaller instances, but merely having a base-case branch does not prove it will be reached.

Computer Programming module

CP22 | Arrays and bounds

A built-in C++ array has a fixed number of same-type elements in contiguous storage: int a[4]={10,20,30,40};.

Valid indices are 0 through 3. a[4] is outside the array; accessing it has undefined behavior.

Java checks bounds and throws ArrayIndexOutOfBoundsException.

Do not predict a C++ out-of-range value from nearby memory.

A standard built-in array bound must be a compile-time constant; variable-length local arrays are compiler extensions. int a[5]={1,2,3}; fills the remaining elements with zeros.

An uninitialized automatic array of int is different. a[i] equals *(a+i), so elements can be accessed using pointer notation without explicitly writing brackets.

Computer Programming module

CP23 | Array algorithms and two-dimensional indexing

For a sum, start total=0 and add each element once.

For a maximum/minimum, initialize from a[0] only after confirming the array is nonempty.

For 50 inputs where nonpositive values are skipped, use a count-controlled loop and either if(x>0) total+=x; or continue for x<=0.

These constructs solve different parts of the task.

int a[R][C] has R rows, each with C elements. a[r][c] uses zero-based row then column.

Its element offset is r*C+c in row-major storage.

In the 4x4 matrix example, a[1][1]=6, a[2][3]=6 and a[3][3]=2.

In Java, length belongs to each array; a traversal ending at arr.length-2 visits fewer elements than a full traversal.

Computer Programming module

CP24 | C strings and C++ string objects

A C string is a character array ending in '\0'. "Hello" needs six char positions; strlen counts five visible characters and excludes the terminator. sizeof a fixed char array reports its allocated bytes, which can exceed strlen+1. cout<<a prints a terminated char sequence, not its numeric address.

<cstring> supplies strcpy(destination,source), strcat(destination,source), strlen and strcmp.

Supply enough destination capacity; strcmp returns zero for equal content, otherwise a sign, not necessarily exactly -1 or 1.

C++ std::string in <string> owns resizable text; + concatenates, == compares content, size()/length() counts stored characters and substr(start,count) extracts a segment.

Standard std::string has no Java-style toUpperCase() or trim() member.

Computer Programming module

CP25 | Pointers, dereferencing and assignment

int x=10; int* p=&x; stores x's address in p. *p reads or writes x; &p is the address of the pointer variable itself.

With p=&a,q=&b, p=q redirects p to b and leaves a and b unchanged.

By contrast *p=*q copies the pointed-to value.

Never dereference an uninitialized, null or dangling pointer.

An array expression commonly converts to a pointer to its first element, but the array is not itself an assignable pointer variable.

A pointer-to-pointer int **pp=&p allows** pp to reach x. A void* can hold an object address but requires conversion to a suitable object pointer before dereferencing in C++.

Computer Programming module

CP26 | Scaled pointer arithmetic and dynamic allocation

If p points inside an array, p+k advances k elements, or k*sizeof(*p) bytes.

Under the explicit assumption sizeof(float)==4 and first address 7000, p+3 has numeric address 7012. cout<<p++ displays the old address then advances p; cout<<*p++ displays the old element then advances p. ++*p changes the element.

Pointer addition p+q is invalid; subtracting pointers within the same array gives an element distance.

Arithmetic must stay within an array or its one-past position; one-past may not be dereferenced.

C++ new T creates a dynamic object, delete releases it; new T[n] pairs with delete[].

Prefer resource-owning standard containers in real programs.

Allocation does not automatically initialize every primitive value unless initialization is requested.

Computer Programming module

CP27 | Structures and member access

A struct groups named members, possibly of different types. struct House{double price; int rooms;}; defines a type; House h{150000,3}; creates an object. h.rooms uses a dot.

House* p=&h; p->rooms is equivalent to (*p).rooms.

The pointer must refer to a live object before member access.

A type definition alone does not create an ordinary instance.

Aggregate initialization assigns members in declaration order.

Modern C++ permits default member initializers such as int rooms=3;.

C++11 list initialization rejects narrowing; a constant floating value exactly representable in the target floating type can be permitted, so 25.00 to float is not necessarily narrowing.

Use 25.0f for clarity and check the language version.

Structures can be copied, passed and returned; a structure cannot contain a full instance of itself, but may contain a pointer to its own type.

Computer Programming module

CP28 | Console input, output and headers

Include <iostream> and qualify std::cin/std::cout or introduce the names explicitly. cout is the standard output stream; << inserts values. cin is standard input; >> extracts formatted values. std::endl inserts a newline and flushes.

C++ names are case sensitive: Cout, Int and std: :cout are not equivalent valid spellings of cout, int and std::cout.

Read a full line with getline(cin,text).

After formatted numeric extraction, consume the pending newline before a line read when required.

Input failure must be tested before trusting a value.

Console output formatting matters: by default a double storing an exact 4 may print 4 rather than 4.0.

Text, spaces and newlines are part of an output trace.

Computer Programming module

CP29 | File streams, modes and safe reading

Include <fstream>. ofstream writes, ifstream reads, and fstream can do both.

The constructor's default ofstream mode argument is ios::out; its ordinary effect is to create/truncate the file.

Distinguish the default mode argument from effective out|trunc behavior. ios::app places each write at the end; ios::ate initially seeks to the end; ios::binary requests binary mode.

Check if(!file) after opening.

Read using while(file>>record) or while(getline(file,line)), because end-of-file is discovered by an attempted read. while(!file.eof()) can process an invalid extra record.

Close or rely on stream destruction to release the resource. "ofstream" is a class, whereas <fstream> is the header.

Computer Programming module

C++ draft: ofstream constructors

CP30 | Mathematical functions and iterative powers

Use <cmath> for sqrt, pow, floor and ceil; integer abs is also provided by <cstdlib>. sqrt(16)=4; pow(3,4)=81; abs(-7)=7. ceil(x) is the least integer-valued result >=x; floor(x) the greatest <=x. Thus ceil(-2.7)=-2 and floor(-2.7)=-3; truncation toward zero differs from floor for negatives.

C++ has no built-in exponentiation operator: ^ is bitwise XOR and ** is not a power operator.

To compute b^e for a nonnegative integer e, start r=1 and repeat r*=b exactly e times; b=4,e=3 gives 64.

Large powers and factorials can overflow fixed-width types; specify a valid input range.

Computer Programming module

CP31 | Correctness by logic, invariants and induction

To prove an implication P⇒Q, a direct proof derives Q assuming P; a contrapositive proves not Q⇒not P; a contradiction assumes the claim false and derives an impossibility.

A single counterexample disproves a universal claim.

Induction proves a base case and proves P(k+1) assuming P(k).

For sum=0,i=1; while(i<=n){sum+=i;++i;}, use invariant sum=(i-1)*i/2.

It holds initially.

Adding i gives i*(i+1)/2, then incrementing i restores the invariant.

Exit gives i=n+1 and sum=n*(n+1)/2.

Termination needs a separate decreasing nonnegative measure n-i+1 while the guard holds.

Assume n>=0 and no arithmetic overflow.

This connects mathematical proof to actual code.

Western University: program verification

Computer Programming module

CP32 | Debugging, errors, exceptions and maintenance

A syntax error violates grammar; a type error can be syntactically well formed but rejected during semantic checking.

A runtime error occurs during execution; a logic error gives the wrong behavior.

A bug is a defect; debugging locates and corrects it.

Read the first useful compiler diagnostic, inspect nearby tokens and declarations, repair and rebuild; a debugger is useful after a runnable program exists.

Test ordinary, boundary and invalid inputs and trace the first point where actual state differs from expected state.

Exceptions transfer control to a matching handler for exceptional conditions; they do not make arbitrary undefined behavior recoverable.

Refactoring improves internal structure while preserving observable behavior.

Version control records changes, supports comparison and rollback; documentation explains use, design and maintenance assumptions.

Computer Programming module

CP33 | Static typing and Java-specific details

Static typing checks types before execution; type inference can still infer a static type, as with C++ auto.

Dynamic typing checks operations using runtime values; it does not mean variables must have explicit static declarations.

Static typing does not always require explicit type annotations.

Java uses boolean, fixed-width numeric primitive types and references to objects.

Java String is immutable, substring(begin,end) uses an exclusive end, equals compares content and == compares references.

Java arrays have a length field.

Scanner nextInt() can leave a newline for nextLine().

C++ std::string is mutable and uses substr(start,count); the two APIs must not be merged.

Computer Programming module

CP34 | Check syntax and trace control flow

A program trace requires the complete code, and a diagram question requires its edges, conditions and labels.

When entry reaches the body rectangle before the decision diamond and a decision branch returns to the body, the body executes before the condition is tested.

This is a post-test do-while loop.

Check that a program is valid before tracing it; state any syntax changes before analyzing the resulting program.

Match pointer-member names exactly: Cout/Cin, C1 and car_mode do not match cout/cin, c1 and car_model.

In the goto sequence above, Year is not printed because control does not reach its label.

With valid C++ syntax for the stated assignments, the trace yields z=5 then a=1.

Printing Z when only z is declared refers to a different, undeclared identifier.

Use <0 for a less-than-zero comparison and the standard int main() declaration for a C++ entry point.

Check int and cout case, quotation marks and comma-expression punctuation before tracing a C++ expression.

Predict output only after the program and its execution conditions are clear.

Computer Programming module

CP35 | Interpret programming questions precisely

Evaluate every choice against the language rules; more than one statement may be valid, or none may be valid.

Distinguish linking from loading.

Distinguish binary representation, machine code and executable files.

Assess class membership and possible initialization separately; both statements can be correct.

Check whether alternative short declarations are equivalent.

Check the C++ version and narrowing rules before evaluating an initialization.

Array subscripting and equivalent pointer access can refer to the same element.

Use the definition of static typing rather than equating it with explicit annotations.

An ordinary integer declaration can be written int x;.

A void function definition can be complete without returning a value; int myFunction(){} can compile, but reaching its end is undefined behavior.

A break statement exits the enclosing loop or switch.

Check required initialization when evaluating const declarations, including alternative const spellings.

Distinguish counting iterations from skipping inputs, and terminating a loop from ending only the current iteration.

After inspecting and changing code, rebuild and retest to complete debugging.

Computer Programming module

CP-S01 | Empty braces initialize aggregate members

For an aggregate with no default member initializers, empty braces initialize its members recursively. Numeric scalar members become zero, and the elements of a character-array member become zero characters. A zero character at the beginning of such an array makes it an empty, terminated C string. This differs from leaving an ordinary automatic scalar or array uninitialized.

C++ working draft - Aggregate initialization

CP-S02 | High-level languages and scripting usage

Python is a high-level programming language with built-in high-level data types. Calling a language a scripting language describes a common way it is used; it does not make that language machine code or assembly. Language level and execution style are different classifications.

Python tutorial - Whetting Your Appetite

CP-S03 | Keywords and names have different roles

A keyword is a reserved word with a predefined meaning in the language grammar. It cannot be repurposed as an ordinary variable or function name. An identifier is a name used to refer to a data object or function; its spelling must obey the identifier-character rules and avoid keywords.

Computer Programming module · page 5

CP-S04 | Java print and println preserve output layout

Java System.out.print writes the supplied value without automatically ending the line. System.out.println writes its value and then a line separator. Successive println calls therefore place their values on separate lines, even when each printed expression also contains a trailing space.

Java SE 17 - PrintStream print and println

CP-S05 | An assignment expression can accidentally become a condition

In C++, simple assignment changes its left-hand object, and the assignment expression refers to that updated object. When an ordinary numeric assignment is used as a condition, the newly assigned value is converted to truth: zero is false and nonzero is true. Equality testing with == instead produces a Boolean result without assigning either operand. Confusing the two can both change program state and choose an unintended branch.

C++ working draft - Assignment and compound assignment operators