hr avatar
Illuminating Talent

Test Results and Interview Guide

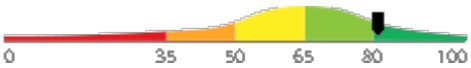
Candidate: **Elizabeth Wantsajob**
Assessment: C++ Programming
Completed: June 24, 2026
Prepared for: Sara Maple
Example Company

What's Included

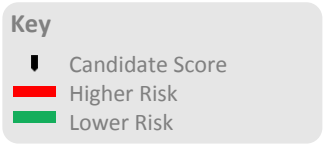
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The C++ Programming assessment measures key factors related to high performance and tenure in this job. Attribute types measured vary by test, but can include cognitive ability, skills, knowledge, personality characteristics, emotional intelligence, and past behavioral history. This report includes a one page summary, followed by detailed results with an embedded interview guide. Note that these results should always be used as a part of a balanced candidate selection process that includes independent evaluation steps, such as interviews and reference checks.

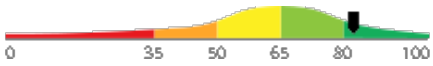
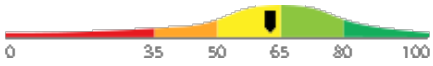
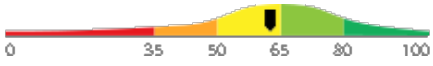
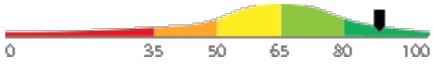
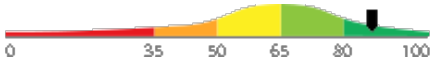
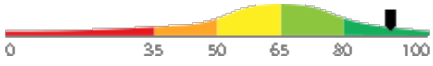
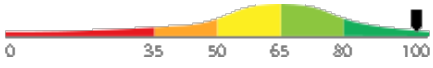
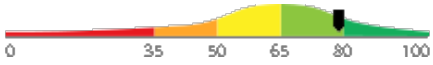
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com C++ Programming June 24, 2026	81	

The candidate demonstrates a comprehensive and advanced mastery of C++ programming, spanning foundational syntax through sophisticated topics such as templates, operator overloading, memory lifecycle management, preprocessor directives, and the Standard Template Library. Knowledge of object-oriented design principles, exception handling, file I/O, and multi-file program organization appears thorough and well-rounded. This individual is well-positioned to independently design, develop, debug, and maintain complex C++ business applications to a high professional standard.



Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Control Flow, Functions, and Program Structure	82	
Memory Management and Pointers (Coding Tasks)	62	
Object-Oriented Programming (OOP) in C++ (Coding Tasks)	62	
Data Types, Variables, and Type System	88	
Error Handling, Debugging, and Exception Management	87	
Memory Management and Pointers	91	
Object-Oriented Programming (OOP) in C++	97	
Standard Template Library (STL)	79	

Comparison

Percentile scores indicate how the candidate compares to other test-takers within various groups. The candidate scored equal to or better than the fraction of test-takers indicated by the percentile.

Test-Taker Group	Percentile
Global	81st
North America	67th
United States	67th
Example Company	75th

Artificial Intelligence (AI) Generated Scores

This table includes one or more scores derived from a large language model AI query. AI-derived scores are non-deterministic. That is, they are not precisely repeatable. Therefore, these scores should always be treated as supplementary information and should never be used exclusively or compared to hard cutoff values.

Estimated Value	Score	Confidence	Interpretation
Knowledge, Skills, and Abilities Summary	-	-	Summary Points (AI): • (Generic Text for Sample Report) Strong performer in Drag and Drop Files tasks, indicating comfort with file management and basic computer interactions. • Demonstrates solid numerical accuracy in Recognizing and Confirming Numbers, a valuable asset in detail-oriented roles. • Moderate overall performance in Analytical Thinking and Attention to Detail, with adequate grammar skills but room for improvement. • Struggles with Reading and Analyzing Problems, which may limit effectiveness in roles requiring critical reading and complex problem-solving. • Lowest performance in Navigating Between Screens, suggesting difficulty with multi-screen software workflows that could impact productivity in computer-intensive roles. Narrative (AI): Elizabeth Wantsajob demonstrates a mixed profile of knowledge, skills, and abilities across the assessed competencies. Elizabeth shows a strong aptitude in Drag and Drop Files, performing well on this technical task and suggesting she is comfortable with this type of computer interaction. This is a notable strength that would translate well into roles requiring file management and basic computer navigation tasks. In the area of Analytical Thinking and Attention to Detail, Elizabeth performs at a moderate level. She demonstrates solid ability in Recognizing and Confirming Numbers, which suggests she is careful and accurate when working with numerical data — a valuable skill in detail-oriented work environments. Her Grammar performance is adequate but leaves room for improvement, indicating she may occasionally make written communication errors. Her weakest area within this competency is Reading and Analyzing Problems, where she struggled to consistently interpret and work through written problem scenarios. This may impact her effectiveness in roles that require critical reading, written comprehension, or complex problem-solving. Elizabeth's most significant area for development is Navigating Between Screens, where she scored considerably lower than the other competencies. This suggests she may have difficulty efficiently moving through software interfaces or multi-screen workflows, which could slow productivity in roles that rely heavily on navigating computer applications or data entry systems. Overall, Elizabeth brings some useful technical strengths, particularly in file management and numerical accuracy, but would benefit from targeted development in software navigation and analytical problem-solving to be fully effective in roles that demand these skills. Computed on: April 2, 2026, 11:09:49PM EDT

Detail

Candidate:	Elizabeth Wantsajob, beth.wantsajob@gmail.com
Assessment:	C++ Programming
Authorized:	June 24, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 24, 2026, 9:57:12PM EDT
Completed:	June 24, 2026, 9:57:12PM EDT
Overall Score:	81

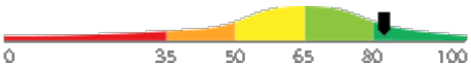
Knowledge and Skills Detail

This section contains a list of job-related knowledge areas and skills that have been evaluated. Low scores in these areas often indicate that additional learning may be required before top performance can be achieved.

Detail | Interview Guide

Control Flow, Functions, and Program Structure

Score: 82



Description:

Covers the use of loops, conditionals, switch statements, and functions including parameter passing, return types, overloading, and scope. Also includes organizing code across multiple files using header files, namespaces, and the build process. These are the everyday building blocks used in every C++ program.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits a strong and comprehensive command of C++ control flow, functions, and program structure, reflecting proficiency across all major topic areas including switch statements, function overloading, scope, namespaces, and multi-file build organization. They are well-equipped to write well-structured, maintainable C++ code using the full range of everyday language building blocks. This level of knowledge is indicative of a highly capable and confident C++ practitioner.

How do you organize a C++ project across multiple files, and what role do header files and namespaces play in keeping your code maintainable?

1	2	3	4	5
Cannot explain the purpose of header files or how to split code across files.		Correctly explains header files and include guards but gives only a basic description of namespaces.		Explains header files, include guards, source file separation, and namespace use with a focus on avoiding naming conflicts and improving maintainability.

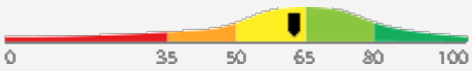
How do you use a function in C++ to avoid repeating the same code in multiple places, and what do you need to include when writing one?

1	2	3	4	5
Cannot describe the components of a function or gives an incomplete or incorrect explanation.		Correctly identifies return type, name, and parameters but does not address scope or reuse clearly.		Clearly explains function structure, discusses parameter passing by value vs. reference, and connects functions to code reuse and maintainability.

Detail

Memory Management and Pointers (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

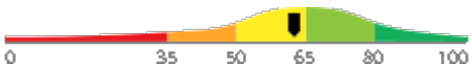
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Object-Oriented Programming (OOP) in C++ (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

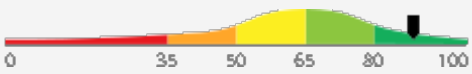
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Data Types, Variables, and Type System

Score: 88



Description:

Covers fundamental and user-defined data types including arrays, structures, enumerations, and type casting, as well as how variables are declared, initialized, and scoped. A solid understanding of C++'s type system is necessary for writing correct, predictable, and efficient code in any application.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate demonstrates a comprehensive and advanced understanding of C++ data types, variables, and the type system, including both fundamental and user-defined types, type casting, and variable scoping. This level of proficiency indicates a strong ability to write efficient, correct, and predictable C++ code and to make well-informed decisions regarding type usage in complex application contexts.

Interview Guide

How do you use structures and enumerations in C++ to represent real-world data in a program, and what advantages do they offer over using plain variables?

1	2	3	4	5
Cannot accurately describe a struct or enum or does not explain their practical benefit.		Correctly defines both and gives basic examples but does not clearly explain the organizational or readability benefits.		Clearly explains both with practical examples, discusses how they improve code clarity and reduce errors, and may mention enum class for type safety.

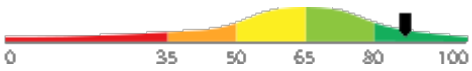
Can you explain the difference between an int, a float, and a char in C++, and describe a situation where choosing the wrong data type could cause a problem in your program?

1	2	3	4	5
Cannot clearly distinguish between the types or does not connect type choice to real consequences.		Correctly defines the types and gives a basic example of a problem but without much practical depth.		Clearly explains the types, discusses overflow, precision loss, or implicit conversion issues, and ties the answer to a realistic programming scenario.

Detail

Error Handling, Debugging, and Exception Management

Score: 87



Description:

Covers how to use try, catch, and throw to handle runtime errors gracefully, interpret compiler and linker errors, and apply debugging techniques to find and fix problems. Writing code that handles errors well and can be debugged efficiently is essential for maintaining reliable business applications.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate demonstrates an advanced and comprehensive mastery of C++ error handling, debugging, and exception management. They are highly proficient in designing robust, exception-safe applications, diagnosing complex compiler and linker errors, and applying sophisticated debugging strategies essential for maintaining reliable, production-quality business software.

Interview Guide

How do you use exception handling in C++ to make sure your program responds gracefully when something goes wrong at runtime, and can you give a real example?

1	2	3	4	5
Cannot accurately describe try/catch or gives an example that does not demonstrate proper usage.		Correctly explains try/catch/throw with a basic example but does not address exception types or cleanup.		Explains exception hierarchy, custom exceptions, and resource cleanup in catch blocks, with a concrete and practical example.

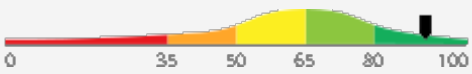
When your C++ program crashes or produces unexpected results, what steps do you take to figure out what went wrong?

1	2	3	4	5
Describes only trial-and-error or cannot name any concrete debugging technique or tool.		Mentions using print statements or a debugger but does not describe a systematic approach.		Describes a clear process including reading error messages, using a debugger like GDB, checking logic, and isolating the problem.

Detail

Memory Management and Pointers

Score: 91



Description:

Covers the use of pointers, references, dynamic memory allocation with new and delete, and avoiding common problems like memory leaks and dangling pointers. Also includes smart pointers from the modern C++ standard library. Effective memory management is critical for writing stable and efficient C++ programs.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits an advanced and comprehensive understanding of C++ memory management and pointers, including dynamic allocation, smart pointers, and strategies for preventing memory leaks and dangling pointers. They demonstrate the depth of knowledge expected to write stable, efficient, and safe C++ code in professional environments. This candidate is well-equipped to handle complex memory management challenges and may serve as a resource for others in this area.

Interview Guide

How do you manage dynamically allocated memory in C++ to avoid memory leaks, and what tools or techniques do you use to catch memory problems?

1	2	3	4	5
Mentions new and delete but cannot explain how to prevent leaks or detect them.		Explains new and delete correctly and mentions smart pointers or basic debugging tools.		Discusses smart pointers (unique_ptr, shared_ptr), RAII principles, and tools like Valgrind or sanitizers with practical context.

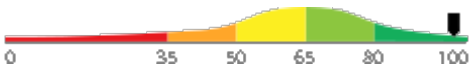
What is a pointer in C++, and can you give an example of when you would use one instead of a regular variable?

1	2	3	4	5
Cannot accurately define a pointer or provides a confused or incorrect example.		Correctly defines a pointer and gives a basic example but does not explain when or why to use one.		Clearly defines a pointer, explains the difference from a reference, and gives a practical use case such as dynamic allocation or passing large objects.

Detail

Object-Oriented Programming (OOP) in C++

Score: 97



Description:

Covers the use of classes, objects, inheritance, polymorphism, encapsulation, and abstraction to design and build programs. Includes constructors, destructors, copy semantics, and object lifecycle management. This is the foundation of most real-world C++ application development.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits a comprehensive and proficient command of C++ programming, encompassing advanced topics such as templates, operator overloading, copy semantics, preprocessor directives, and object lifecycle management, in addition to strong foundational skills. They are well-equipped to independently develop, debug, deploy, and maintain complex C++ business applications while adhering to professional standards for code quality and maintainability.

Interview Guide

Describe a situation where you used inheritance and polymorphism together in a C++ program. What problem did it solve and how did you implement it?

1	2	3	4	5
Cannot describe a real example or confuses inheritance with other concepts.		Gives a valid example of inheritance but explains polymorphism only at a surface level.		Clearly explains both concepts with a concrete example, including virtual functions and how the design improved the program.

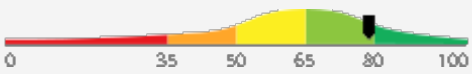
Can you explain what a class is in C++ and describe how you would use one in a program you are building?

1	2	3	4	5
Vague or incorrect description of a class; cannot connect it to practical use.		Correctly defines a class and gives a basic example but lacks detail on members or methods.		Clearly defines a class, explains access specifiers, and describes a practical use case with methods and data members.

Detail

Standard Template Library (STL)

Score: 79



Description:

Covers the use of STL containers such as vector, map, and set, along with iterators and built-in algorithms like sort, find, and transform. The STL provides ready-made, efficient tools that are used constantly in everyday C++ development to handle data storage, retrieval, and manipulation.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate exhibits a solid and competent understanding of the STL, including proficient use of containers, iterators, and standard algorithms in everyday C++ development. They are capable of leveraging STL tools effectively for most data storage, retrieval, and manipulation tasks, with some room to deepen expertise in advanced areas.

Interview Guide

Walk me through how you would choose between different STL containers like a vector, map, or set when designing a feature in a C++ application.

1	2	3	4	5
Cannot distinguish between containers or applies them incorrectly to use cases.		Correctly identifies general use cases for each container but does not discuss performance trade-offs.		Clearly explains the trade-offs in access speed, ordering, and memory, and ties choices to specific, practical scenarios.

What is a vector in C++, and how would you use it to store and access a list of items in a program?

1	2	3	4	5
Cannot accurately describe a vector or confuses it with a plain array.		Correctly describes a vector and demonstrates basic usage such as push_back and indexing.		Describes a vector clearly, explains its advantages over arrays, and demonstrates iteration or use with STL algorithms.

IT Coding Tasks

During the assessment, the candidate was asked to write one or more programs or scripts. Their responses are included below for review.

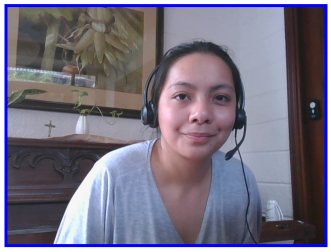
Question or Task	Response

Complete the provided partial C program by filling in the missing sections marked with TODO comments. Your completion must use standard C keywords and libraries.
 	
The program declares a function called duplicate_array that:
1. Takes a const int pointer to a source array and its length as parameters.
2. Uses calloc to allocate a new int array of the same length.
3. Returns NULL if calloc fails.
4. Copies each element from the source array into the new array using pointer arithmetic (not array subscript notation).
5. Returns the pointer to the newly allocated copy.
 	
In main, the program:
1. Declares and initializes a stack array of 4 integers with values 5, 15, 25, 35.
2. Calls duplicate_array to create a heap-allocated copy.
3. Checks for NULL and prints an error and returns 1 if the call failed.
4. Prints each element of the duplicate using a loop.
5. Frees the duplicate array.
 	
Use appropriate indentation, common C coding conventions, and add brief inline comments where needed. Type your completed source code as your response.
	#include <stdio.h>
	#include <stdlib.h>
	
	int *duplicate_array(const int *src, int length) {
	/* TODO: Use calloc to allocate a new array of 'length' integers, return
	NULL if calloc fails, copy elements from src using pointer arithmetic,
	and return the new pointer. */
	calloc(303);
	}
	
	int main(void) {
	/* TODO: Declare and initialize a stack array of 4 integers: 5, 15, 25, 35,
	then call duplicate_array and store the result. Check for NULL and
	print an error message returning 1 if it failed. */
	array[4]={5,15,25,35};
	
	int i;
	
	/* Print each element of the duplicate */
	for (i = 0; i < 4; i++) {
	printf("duplicate[%d] = %d\n", i, *(duplicate + i));
	}
	
	/* Free the duplicate array */
	free(duplicate);
	return 0;
	}
	
	Comments (AI): The code segment has several syntax errors and incomplete implementation. The duplicate_array function does not correctly allocate memory or copy elements. The main function has syntax errors and does not properly call the duplicate_array function. However, the structure and intent of the code are somewhat clear, and the code attempts to follow the requirements.

Identity Confirmation Photos

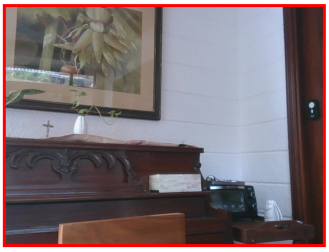
The following photos of the candidate and any identification were uploaded during the assessment session.

Photo Analysis Results	
- Risk:	Medium risk of cheating based on image inconsistencies
- Percent match among processed faces	100%
- Total images processed	17
- Total images with valid faces	14 (82%)
- Total pairs of faces compared	13
- Pairs in which faces matched	13 (100%)

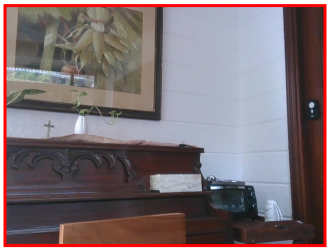


Pre/Post-Test Photo

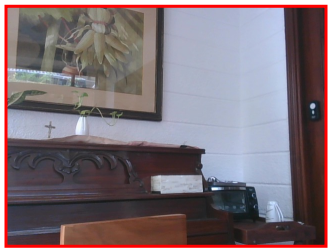
ID Photo



In-Test Error Detected (No Face Detected)

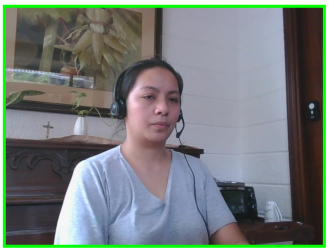


In-Test Error Detected (No Face Detected)

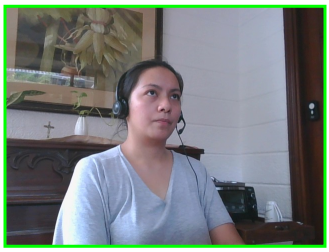


In-Test Error Detected (No Face Detected)

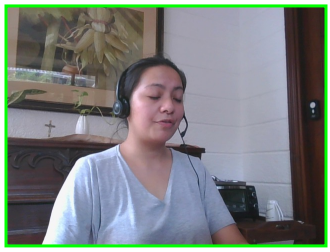
In-Test Photo



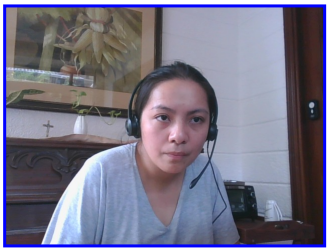
In-Test Photo



In-Test Photo



In-Test Photo



Pre/Post-Test Photo

Resume or CV

Summary Updated on

Motivated career professional with extensive experience in office administration and management. Proven track record of improving efficiency, reducing costs, and enhancing office operations through strategic initiatives and technology implementation.

Objective

I am seeking a role where I can use my many skills and my exceptional judgment and empathy for customers to make a difference to a growing company.

Education

- Associate of Applied Science in Office Administration, Portland Community College, 2020

Experience

- General Office Clerk, Paramount Office Management, 09/2023 – Present
- Administrative Assistant, Global Enterprises Inc., 04/2021 – 08/2023
- Administrative Assistant, Innovative Business Solutions Ltd., 07/2019 – 03/2021

Other Qualifications

- Microsoft Office Specialist (MOS) Certification
- Certified Administrative Professional (CAP)
- International Association of Administrative Professionals (IAAP) Certification

Report Preparation Notes

- Hiring decisions should never be based on a single source of information. The most effective use of this assessment report is as a part of a multi-faceted program of candidate evaluation that includes resume review, interviews, and reference checks.
- Overall vs Percentiles Scores: The overall score reflects the success in the test, based on the mean (average) and standard deviation of the test scores. The percentile score reflects the percentage of test-takers who scored equal or below this overall score. We recommend you use the Overall Score as your primary evaluation criteria. However, percentile scores can often be useful in comparing specific candidates against one another and with a group, such as for test takers in a certain organization or within a certain account.
- Note that comparison information is calculated based on completed instances of this assessment at that time the assessment is scored. As additional instances are completed, the comparative data may change. You can always update a report to the current values by clicking on 'Recalculate Percentiles' within the online results viewing pages at www.hravatar.com.
- Most competency scores are norm-based, which means that they can be interpreted in terms of their distance from the average or mean score. For all scales, a score equal to the mean receives a score of 65 and scores above and below this value are set so that a score change of 15 equals one standard deviation.
- For linear competencies, higher is better across the entire scale. For these scales a score between 65 and 80 (light green) represents 0 to 1 standard deviation above the mean and a score above 80 (dark green) represents more than one standard deviation above the mean. Similarly, a score of 50 - 65 (yellow) represents 0 to 1 standard deviation below the mean, while a score of 35 - 50 (orange) equates to 1 to 2 standard deviations below the mean, and a score below 35 represents more than 2 standard deviations below the mean.
- Sim ID: 20662-1, Key: 0-0, Rpt: 68, Prd: 9540, Created: 2026-06-24 21:57 EDT
- UA: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; Touch; rv:11.0) like Gecko

Score Calculation Detail

The following table provides a summary of how the overall score was calculated from each of the individual competency scores. First, all competency scores are calculated on a scale of 0-100. Note that some competencies use their color category rather than their actual numeric score in the overall calculation. For these, a standard score associated with the assigned color category is used in the overall score calculation rather than the actual numeric score. This is reflected in the ''Score Value Used'' column. Next, a weighted average of scores is computed using individual competency weights, typically set using job analysis data provided by the US Government Occupational Information Network (O*Net).

Competency	Score	How applied to overall	Score Value Used	Weight (%)
Control Flow, Functions, and Program Structure	82.4571	Numeric Score	82.4571	12.5000
Data Types, Variables, and Type System	88.8100	Numeric Score	88.8100	12.5000
Error Handling, Debugging, and Exception Management	87.0560	Numeric Score	87.0560	12.5000
Memory Management and Pointers	91.2604	Numeric Score	91.2604	12.5000
Memory Management and Pointers (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Object-Oriented Programming (OOP) in C++	97.7362	Numeric Score	97.7362	12.5000
Object-Oriented Programming (OOP) in C++ (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Standard Template Library (STL)	79.3299	Numeric Score	79.3299	12.5000
Weighted Average:				81.5758
Final Overall Score:				81

Notes

(This area is intentionally blank - it's reserved as space for your notes.)