C++ Programming Guide

# Getting Started

**Compiling and Running Your Code**

**Basic compilation:**

bash

g++ filename.cpp

./a.out

**Compilation with custom name:**

 Use the terminal (command line) for both commands

bash

g++ filename.cpp -o myprogram

./myprogram

 First command compiles your code into machine language  Second command runs the compiled program

# Basic Output

**Printing to Console**

cpp

std::cout << "Hello World!\n";

 This is how you display text on screen

 Always end statements with semicolon ( ) creates a new line



;



\n

**Chaining Output**

cpp

std::cout << "Your score is: " << score << "\n";

Use to connect multiple pieces of output

<<

# Comments

**Single-line Comments**

cpp

*// This is a comment*

**Multi-line Comments**

cpp

*/\* This is a*

*multi-line comment \*/*

# Variables

**Variable Types**

 **int** - Whole numbers (1, 42, -5)

 **double** - Numbers with decimals (3.14, -2.5)

 **char** - Single character ('A', '7', '$')  **float** - Same as double but

lesser precision and lesser memory used compared to double (Needs to state like this 1.2f, 42.54F, otherwise it will be treated as a double.

 **string** - Text ("Hello", "C++ is fun")  **Long**- 90000L Needs to be stated like

so, for it to be read as a Long variable rather than an int variable which has a lesser value range than a long variable

 **bool** - True or false values

**Creating Variables**

cpp

int age; *// Declare without value* int score = 100; *// Declare with value* double price = 29.99;

string name = "Alice";

bool isStudent = true;

**Checking Variable Values**

cpp

std::cout << score << "\n";

**Constants**

cpp

const double PI = 3.14159;

 Values that cannot be changed after creation

# Math Operations

**Basic Operators**

 **+** Addition

**-** Subtraction

**\*** Multiplication

 **/** Division

 **%** Modulo (remainder after division)

**Power Function**

cpp

pow(number, exponent) *// Example: pow(2, 3) = 8*

**User Input**

cpp

int userAge;

std::cout << "Enter your age: "; std::cin >> userAge;

**Type Conversion**

cpp

int wholeNumber = 42; double decimalNumber;

decimalNumber = (double) wholeNumber; *// Converts int to double*

**Comparison Operators**

 **==** Equal to

 **!=** Not equal to

**>** Greater than

 **<** Less than

 **>=** Greater than or equal to

 **<=** Less than or equal to



*=*

*Note: Use*

*==*

*for comparison,*

*for assignment*

# Conditional Statements

**If-Else Structure**

cpp

if (condition) {

*// Code if condition is true*

} else if (another\_condition) {

*// Code if second condition is true*

} else {

*// Code if no conditions are true*

}

**Switch Statement**

cpp

switch (variable) { case value1:

*// Code for value1*

break; case value2:

*// Code for value2*

break; default:

*// Code if no cases match*

}

# Logical Operators

 **&&** AND (both conditions must be true)

 **||** OR (at least one condition must be true)

 **!** NOT (reverses true/false)

cpp

if (age >= 18 && hasLicense) {

*// Both conditions must be true*

}

if (isWeekend || isHoliday) {

*// At least one condition must be true*

}

if (!isRaining) {

*// True when it's NOT raining*

}

# Loops

**For Loop**

cpp

for (int i = 0; i < 10; i++) {

*// Code repeats 10 times*

*// i starts at 0, continues while i < 10, increases by 1 each time*

}

**While Loop**

cpp

while (condition) {

*// Code repeats as long as condition is true*

}

# Data Storage

**Vectors (Dynamic Arrays)**

Vectors are dynamic arrays that can grow and shrink during program execution. Unlike regular arrays, vectors can change size and provide many useful functions.

**Creating and Initializing Vectors**

cpp

std::vector<int> numbers = {1, 2, 3, 4, 5}; *// Initialize with values* std::vector<int> empty\_vector; *// Empty vector* std::vector<double> decimals(10); *// Vector with 10 default values*

std::vector<string> names(5, "Unknown"); *// 5 elements, all set to "Unknown"*

**Basic Vector Operations**

**Adding and Removing Elements:**

cpp

numbers.push\_back(6); *// Add element at the end*

numbers.emplace\_back(7); *// Construct and add element at the end (more efficient)*

numbers.pop\_back(); *// Remove the last element*

numbers.clear(); *// Remove all elements*

**Accessing Elements:**

cpp

std::cout << numbers[0]; *// Access first element (no bounds checking)* std::cout << numbers.at(0); *// Access first element (with bounds checking)* std::cout << numbers.front(); *// Access first element*

std::cout << numbers.back(); *// Access last element*

**Size and Capacity**

std::cout << numbers.size();

*// Number of elements currently in vector*

std::cout << numbers.capacity(); *// Total space allocated* std::cout << numbers.empty(); *// Returns true if vector is empty* numbers.resize(10); *// Resize to contain 10 elements*

numbers.reserve(20); *// Reserve space for at least 20 elements*

numbers.shrink\_to\_fit(); *// Reduce capacity to fit current size*

cpp

**Advanced Vector Operations Inserting and Erasing:**

cpp

*// Insert element at specific position*

numbers.insert(numbers.begin() + 2, 100); *// Insert 100 at index 2*

*// Construct and insert at specific position*

numbers.emplace(numbers.begin() + 1, 50); *// Insert 50 at index 1*

*// Remove element at specific position*

numbers.erase(numbers.begin() + 2); *// Remove element at index 2*

*// Remove range of elements*

numbers.erase(numbers.begin() + 1, numbers.begin() + 3); *// Remove elements from index 1 to 2*

**Assignment and Swapping:**

cpp

std::vector<int> other\_vector;

numbers.assign(5, 100); *// Assign 5 copies of value 100*

numbers.swap(other\_vector); *// Swap contents with another vector*

**Iterators (Advanced)**

Iterators are like pointers that help you navigate through vector elements:

*// Forward iterators*

for (auto it = numbers.begin(); it != numbers.end(); ++it) { std::cout << \*it << " ";

}

*// Reverse iterators*

for (auto it = numbers.rbegin(); it != numbers.rend(); ++it) { std::cout << \*it << " ";

}

*// Constant iterators (read-only)*

for (auto it = numbers.cbegin(); it != numbers.cend(); ++it) { std::cout << \*it << " "; *// Can't modify elements*

}

cpp

**Complete Vector Method Reference**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | | **Description** | **Example** |
| push\_back(value) | | Add element at the end | vec.push\_back(5); |
| pop\_back() | | Remove last element | vec.pop\_back(); |
| size() | | Number of elements | vec.size() |
| empty() | | Check if vector is empty | vec.empty() |
| clear() | | Remove all elements | vec.clear(); |
| resize(n) | | Resize to n elements | vec.resize(10); |
| capacity() | | Total allocated space | vec.capacity() |
| reserve(n) | | Reserve space for n elements | vec.reserve(100); |
| shrink\_to\_fit() | | Reduce capacity to size | vec.shrink\_to\_fit(); |
| front() | | Access first element | vec.front() |
| back() | | Access last element | vec.back() |
| at(index) | | Access with bounds checking | vec.at(0) |
| operator[] | | Access without bounds checking | vec[0] |
| insert(pos, value) | | Insert at position | vec.insert(vec.begin(), 5); |
| erase(pos) | | Remove at position | vec.erase(vec.begin()); |
| erase(start, end) | | Remove range | vec.erase(vec.begin(), vec.end()); |
| assign(n, value) | | Assign n copies of value | vec.assign(5, 100); |
| swap(other) | | Swap with another vector | vec.swap(other\_vec); |
| emplace\_back(value) | | Construct and add at end | vec.emplace\_back(5); |
| emplace(pos, value) | | Construct and insert at position | vec.emplace(vec.begin(), 5); |
| C | |  | C |
| **Iterator Methods**  **Method** | **Description** | | | |
| begin() | Iterator to first element | | | |
| end() | Iterator past last element | | | |
| rbegin() | Reverse iterator to last element | | | |
| rend() | Reverse iterator before first element | | | |
| cbegin() | Constant iterator to first element | | | |
| cend() | Constant iterator past last element | | | |
| crbegin() | Constant reverse iterator to last element | | | |
| crend() | Constant reverse iterator before first element | | | |

**Practical Example**

cpp

#include <iostream> #include <vector>

int main() { std::vector<int> scores;

*// Add some scores* scores.push\_back(85); scores.push\_back(92); scores.push\_back(78);

std::cout << "Number of scores: " << scores.size() << "\n"; std::cout << "First score: " << scores.front() << "\n"; std::cout << "Last score: " << scores.back() << "\n";

*// Print all scores*

for (int i = 0; i < scores.size(); i++) {

std::cout << "Score " << i + 1 << ": " << scores[i] << "\n";

}

return 0;

}

**Arrays (Fixed Size)**

cpp

int scores[5];

*// Array of 5 integers*

int grades[] = {90, 85, 92, 78}; *// Array with initial values*

 Size cannot be changed after creation  Values can be modified

# Functions

**Creating Functions**

cpp

**Using Functions**

*// Function that doesn't return anything*

void greetUser() { std::cout << "Hello!\n";

}

*// Function that returns a value*

int addNumbers(int a, int b) { return a + b;

}

cpp

greetUser();

*// Call function with no parameters*

int result = addNumbers(5, 3); *// Call function with parameters*

**Parameters vs Arguments**

 **Parameters**: Variables in function definition ( )

int a, int b

 **Arguments**: Actual values passed to function ( )

5, 3

# Common Error Types

1. **Compile-time errors**: Syntax mistakes found when compiling
2. **Link-time errors**: Problems connecting different parts of program
3. **Run-time errors**: Errors that occur while program is running
4. **Logic errors**: Program runs but gives wrong results

# Key Tips

Always end statements with semicolon ( ) Use meaningful variable names



;

Indent your code for readability Test your code frequently

Start with simple programs and gradually add complexity