

Compile time layout Demonstration

Source files (.cpp/.CC/.C, .h)
Intermediate Files
Object files (.obj)
Internals Executable File Format
Compile time v/s link time v/s Runtime
Preprocessor Definitions/ Compile time switches
Building on GNU tools

Examine Runtime Application's memory layout

Code Segment
Data Segment
Stack, Heap, BSS, data
CPU Registers
Static link library v/s Dynamic link library

C++ Object Implementation

How a compiler interprets Object
Various Object Model
Simple/Table driven object model
Special member functions
Constructors, Destructors, Copy Constructor, Assignment Operator
When Are User-Written Copy Constructors And Assignment
Implementing Copy Constructor and Assignment Operator
Blocking Object Copying
Constructors and Destructors Should Be Minimal

Object Initialization & Cleanup

Compiler Synthesized Constructor & Destructor
Deep copy v/s Shallow copy
explicit constructor
Copy Constructor v/s Assignment operator
Initialization v/s Assignment
Order of Initialization

Dynamic Memory Management

Types of Storage
POD (Plain Old Data) and non-POD Objects
The Lifetime of a POD Object
The Lifetime of a non-POD Object
Allocation and De allocation Functions
malloc() and free() Versus new and delete
Exceptions during Object Construction
Alignment Considerations

The Size Of A Complete Object Can Never Be Zero
User-Defined Versions of new and delete Cannot Be Declared in a Namespace
Overloading new and delete in a Class
Guidelines for Effective Memory Usage
Explicit Initializations of POD Object
Data Pointers Versus Function Pointers
The const and volatile Properties of an Object

Class Members

Singleton Classes

Virtual Functions

Internals
Dual Dispatching
Casting Internals
Object Slicing
Virtual Inheritance

Runtime type identification

Introduction
Structure Of This Chapter
Making Do Without RTTI
RTTI constituents
The Cost of Runtime Type Information

Templates Internals

Generic class
Class Template Full specialization
Class Template Partial specialization
Template with friend, inheritance

Optimizing your code

Introduction
Before Optimizing Your Software
Declaration Placement
Inline Functions
Optimizing Memory Usage
Speed Optimizations
A Last Resort

C Language compatibility issues

Introduction

Differences Between ISO C and the C Subset of ANSI/ISO C++
Quiet Differences Between C and C++
Migrating From C to C++
Designing Legacy Code Wrapper Classes
Multilingual Environments
C and C++ Linkage Conventions
Minimize the Interface Between C and C++ Code
Mixing `<iostream>` Classes with `<stdio.h>` Functions
Accessing a C++ Object in C Code

Fitting C++ in Embedded System Softwares

Dos and Donts
Runtime Considerations
Template Considerations
Impact of Other Features of C++
Namespace
Template
RTTI
Exception handling

Deprecated Features in C++

Use of an Operand of Type `bool` with the Postfix `++` Operator
Use of `static` to Declare Objects in Namespace Scope
Access Declarations
Implicit Conversion from `const` to non-`const` Qualification for String Literals
Standard C Headers
Implicit `int` Declarations
Other Deprecated Feature
Conclusion