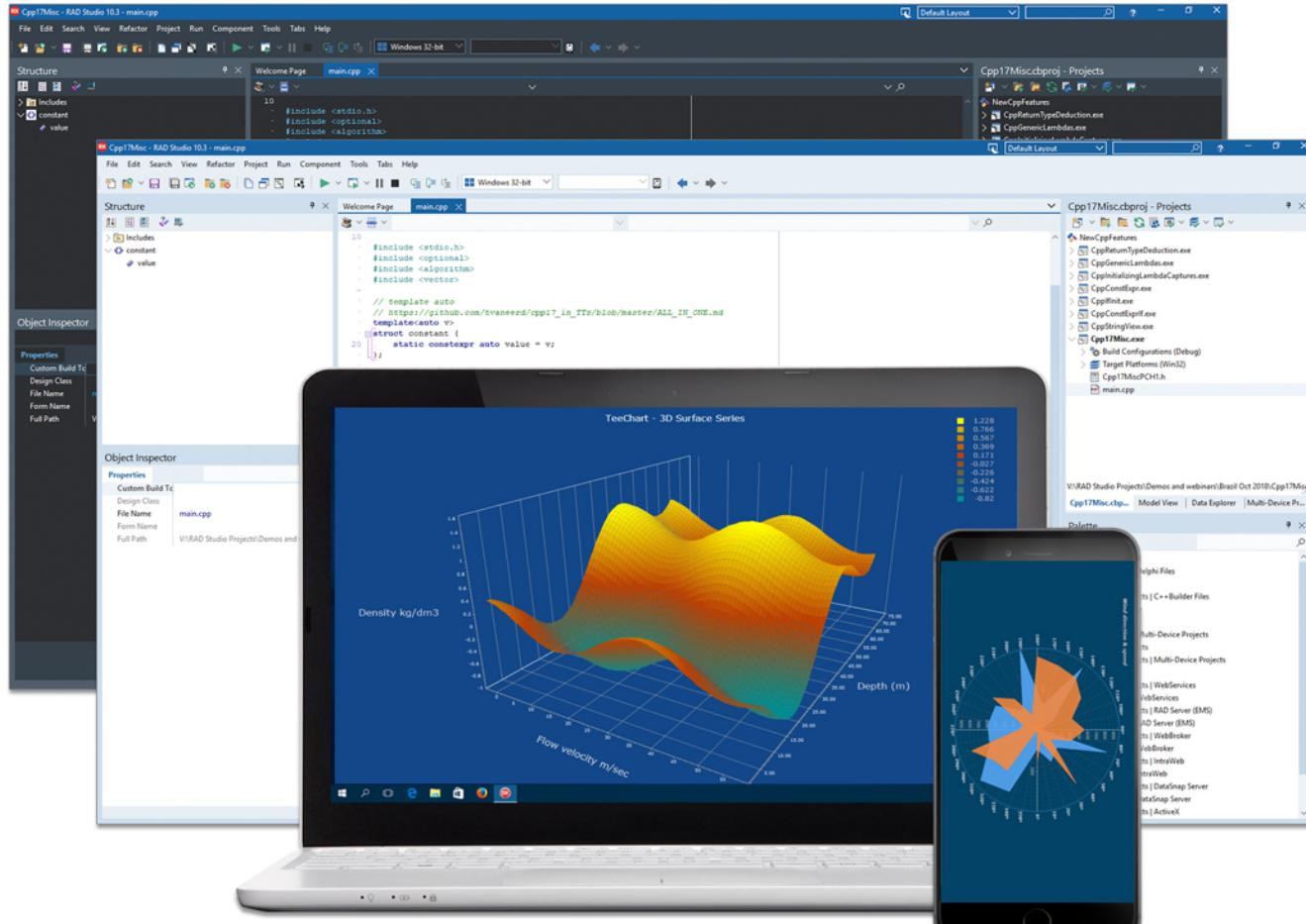


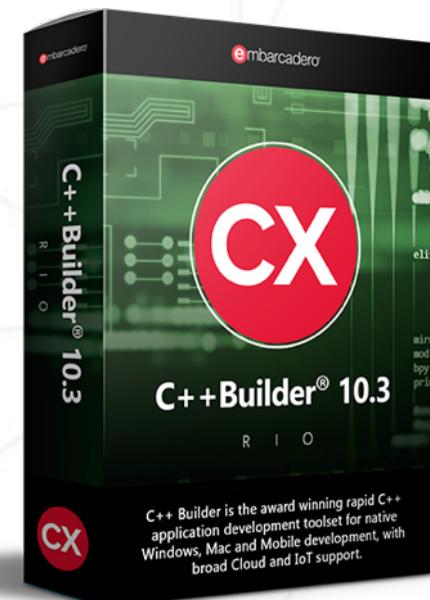
# embarcadero® C++ Builder



embarcadero®



Al Mannarino  
Embarcadero Engineer



C++ Builder is the award winning rapid C++ application development toolset for native Windows, Mac and Mobile development, with broad Cloud and IoT support.

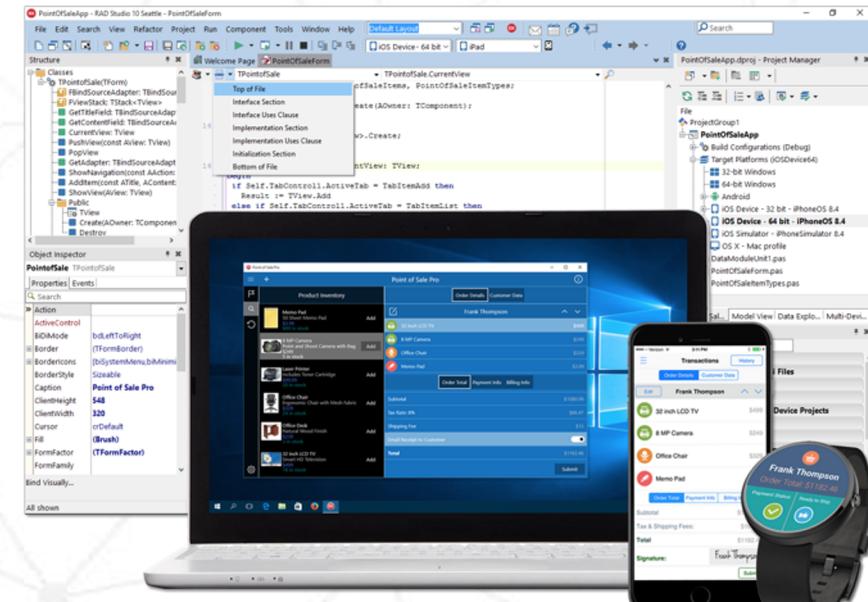
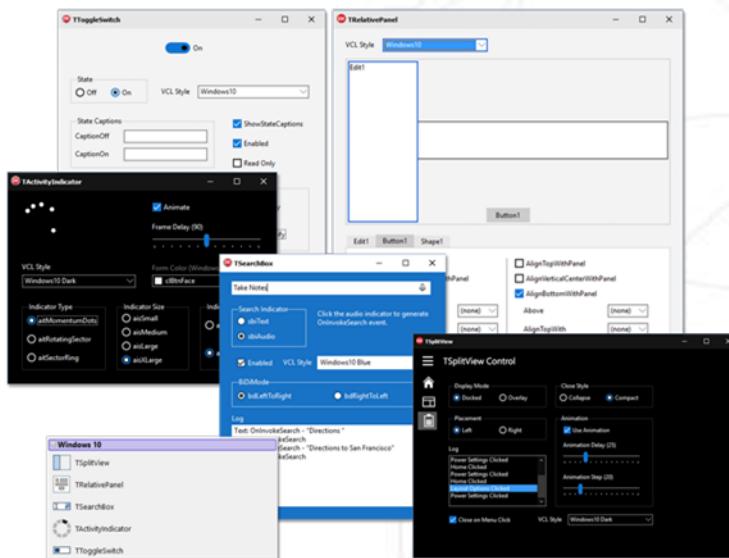
# Agenda



## 1. C++ Builder Overview

## 2. C++ Builder Levels of Development:

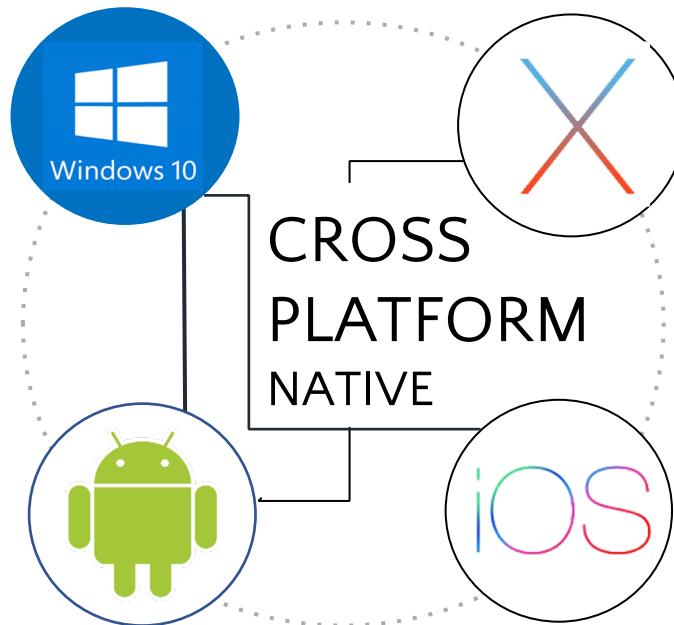
1. Components ( Visual Component Library (VCL) / FireMonkey (FMX) )
2. Common Libraries (Run Time Library (RTL))
3. Platform APIs ('Touch the Metal')



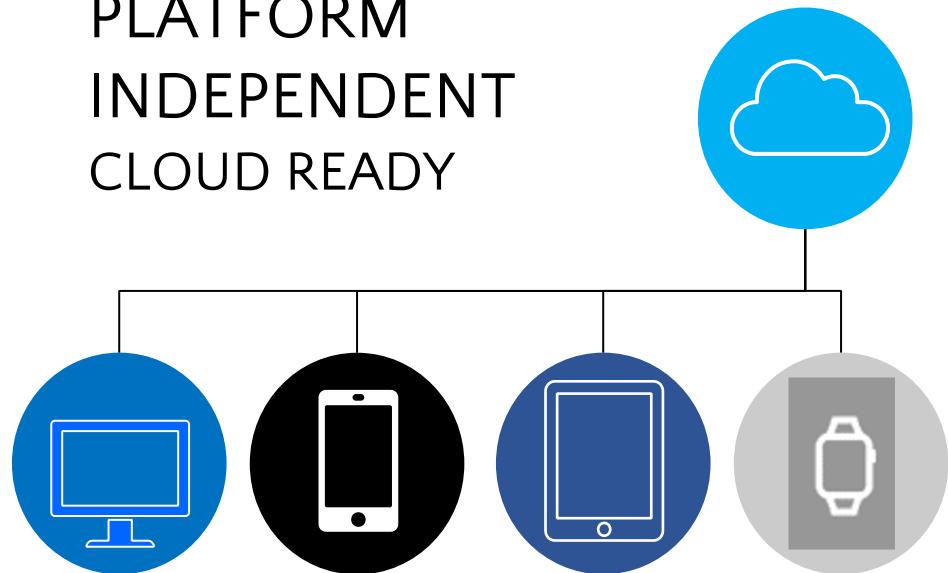
# What is C++ Builder?



C++ Builder is both a **C++ IDE** plus a **Full Stack** for application and server development, with features C++ developers love: **design, code, debug, and test for cross-platform deployment with native performance.**



PLATFORM  
INDEPENDENT  
CLOUD READY



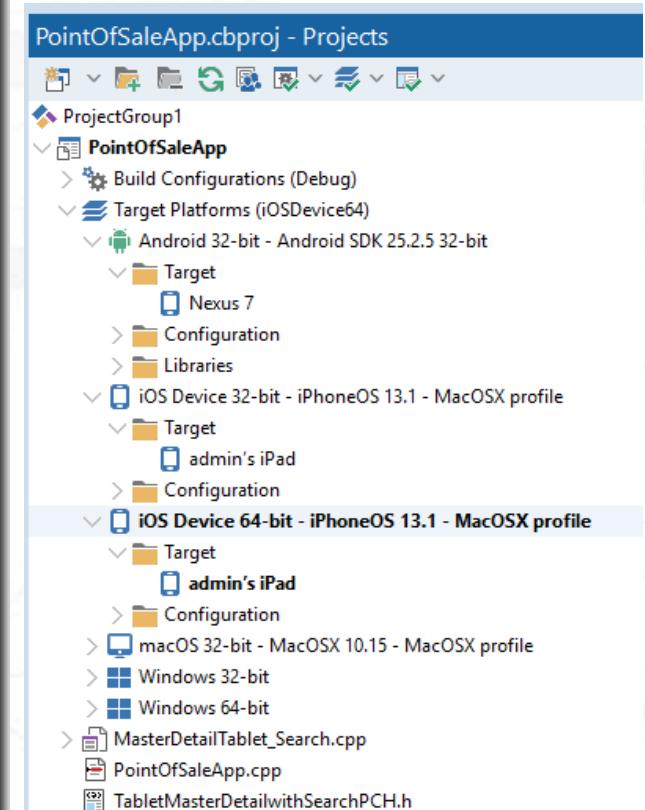
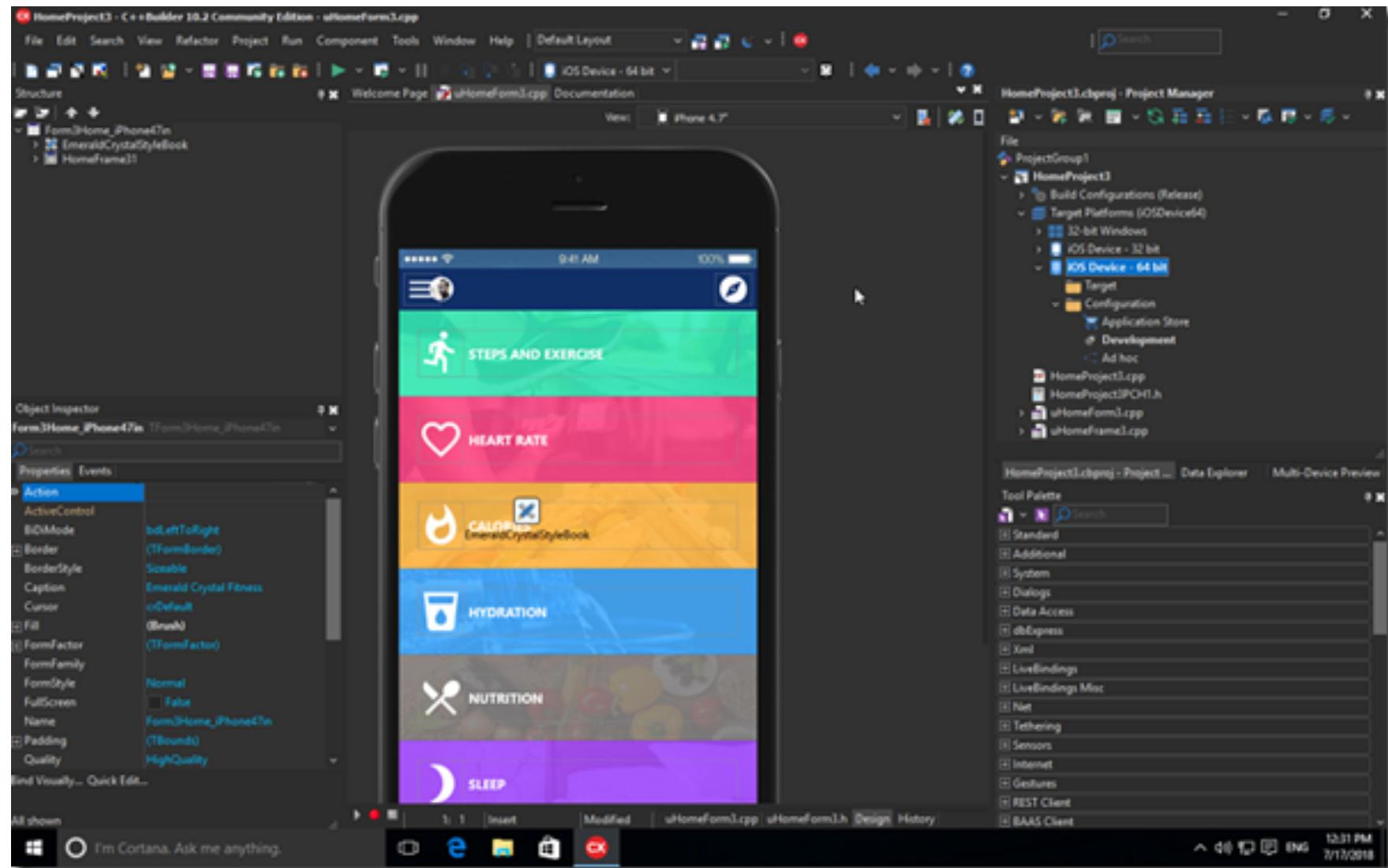


# Components - Visual Component Library (VCL)

The screenshot displays a software development environment for a Visual Component Library (VCL) application. The interface includes:

- Left Panel (uMain.cpp):** Shows a list of components categorized by location and motion. Components include: IsLocation, tMotion, msAccelerometr, tOrientation, osInclinometer, lAccel, lCompass, and osCompass.
- Center Panel:** A window titled "Manage" containing a button labeled "Turn On".
- Right Panel (VCL Sensors C++ Projects):**
  - Projects:** VCLSensorsCpp.cbproj - Projects
  - Toolbars:** Standard, Additional, Samples, TeeChart Std, IntraWeb Standard, QReport Web, Raize Edits, Raize Buttons.
  - Search Bar:** A search bar with the placeholder "button".
  - Component Tree:** ProjectGroup1, VCLSensorsCpp.exe, Build Configurations (Debug), Target Platforms (Win64), Windows 32-bit, Windows 64-bit, uMain.cpp, VCLSensorsCpp.cpp, VCLSensorsCppPCH1.h.
- Bottom Navigation:** Includes icons for back, forward, and search, along with tabs for "1: 1", "Insert", "uMain.cpp | uMain.h", "Design" (which is selected), and "History".

# Components – Multi-Device (Firemonkey or FMX)

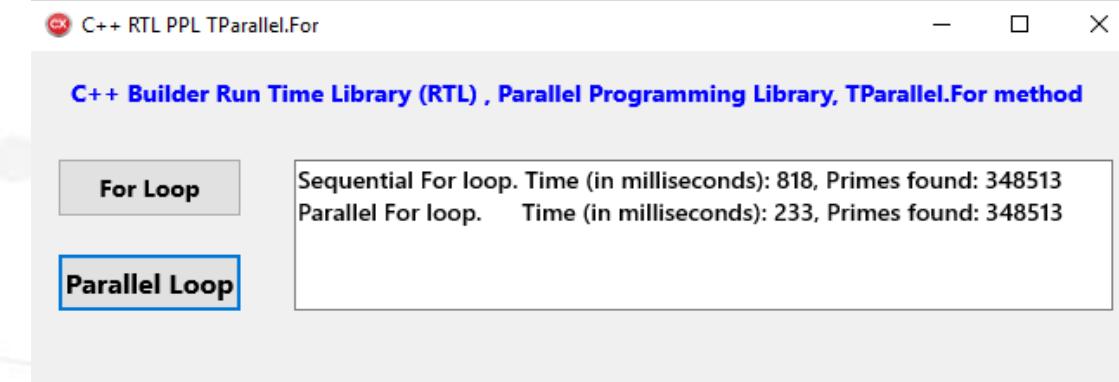




# Common Libraries (RTL)

```
uMain.cpp X System.Threading.hpp |  
1 // -----  
2 //-----  
3 //-----  
4 //-----  
5 #include <fmx.h>  
6  
7 #include <System.Threading.hpp> // TParallel::For  
8 #include <System.Diagnostics.hpp> // System::Diagnostics::TStopwatch  
9 #include <System.SyncObjs.hpp> // TInterlocked::Increment  
10  
11 #pragma hdrstop  
12  
13 #include "uMain.h"  
14 //-----  
15 #pragma package(smart_init)  
16 #pragma resource "*.*.fmx"  
17 TForm2 *Form2;  
18  
19 //-----  
20 //-----  
21 fastcall TForm2::TForm2(TComponent* Owner) : TForm(Owner) {  
22 }  
23  
24 //-----  
25 bool __fastcall TForm2::IsPrime(int N) {  
26     int Test, k;  
27     bool aPrime;  
28     if (N <= 3) {  
29         return N > 1;  
30     }  
31     else if (((N % 2) == 0) || ((N % 3) == 0)) {  
32         return false;  
33     }  
34     else {  
35         TParallel::For(1, N - 1, ForStep)  
36             if (Test % k == 0) {  
37                 aPrime = false;  
38                 break;  
39             }  
40         if (aPrime) {  
41             return false;  
42         }  
43     }  
44 }
```

```
#include <System.Math.hpp>  
#include <System.Bluetooth.hpp>  
#include <System.Sensors.hpp>  
#include <System.Threading.hpp>
```





# Platform APIs

```
#include <fmx.h>
#if defined(TARGET_OS_IPHONE) || defined(TARGET_IPHONE_SIMULATOR)
#include <iOSapiUIKit.hpp>
#endif
#pragma hdrstop

#include "uMain.h"
#if defined(__ANDROID__)
#include <Androidapi.JNI.Os.hpp>
#include <Androidapi.Helpers.hpp>
#endif
//
```





# New C++ Builder 10.4 Features

# New Windows 64-bit C++ Debugger

New modern LLDB based debugger for Windows

- Address key C++ debugging scenarios, stability, and features
- Better evaluation and inspection
- Includes ‘formatters’:
  - Allows you to evaluate and inspect complex types, including STL types (vector, deque, map etc), strings, and even your own custom complex types
- New foundation for future C++ debugger extensions

The screenshot illustrates the new Windows 64-bit C++ Debugger interface. On the left, a code editor displays C++ code with a breakpoint at a specific line. A tooltip shows the current state of a variable named 'myMap'. On the right, a 'Local Variables - Thread 4' window lists various variables and their values.

Code in the editor:

```
std::map<int, std::string> myMap {  
    { 0, "zero" }, { 1, "one" }, { 1000, "pelagic argosy" }  
};  
myMap[99] = "Brooklyn";  
if (myMap.size() > 3) {  
}  
    myMap | size=4  
}  
    > [0] | ...  
    > [1] | ...  
    > [2] | ...  
    > first | 99  
    > second | "Brooklyn"
```

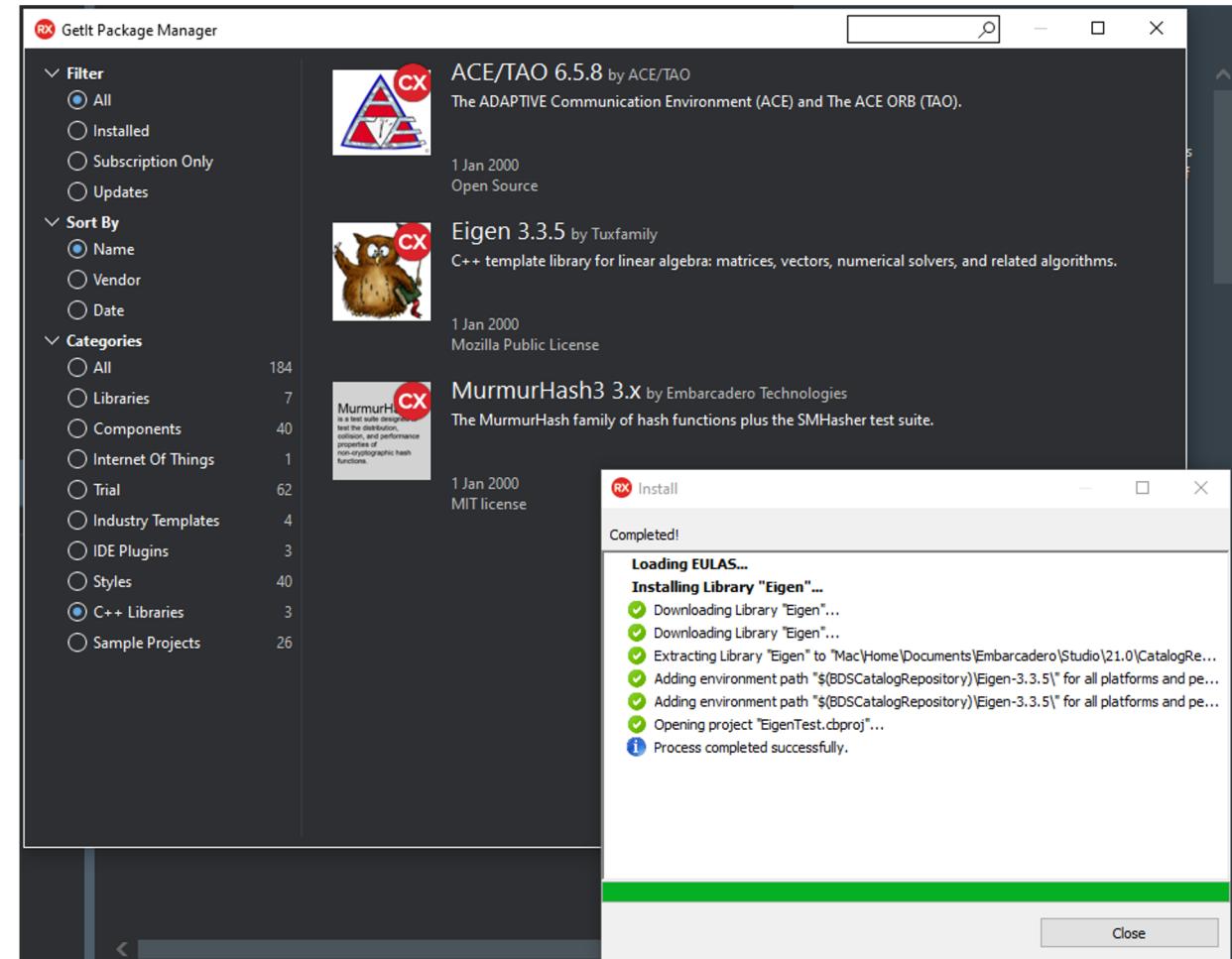
Local Variables - Thread 4

Name	Value
> stdstring	"std::string evaluation - demo"
> stdwstring	"std::wstring evaluation"
> delphiString	"Delphi String evaluation"
> utf8String	"Delphi UTF8 string evaluation"
mySharedPtr	{...}
_Ptr	0x0000000003bb1130
> m_str	"demo structure"
m_i	999
_Uses	1
_Weak	1

# Expanded C++ Libraries Support

Support for common useful open source C++ libraries

- Several new key libraries can be downloaded via the GetIt Package Manager:
  - libSIMDpp
  - NemaTode
  - SDL
  - and others
- Boost
- CMake and RTL work to improve library compatibility



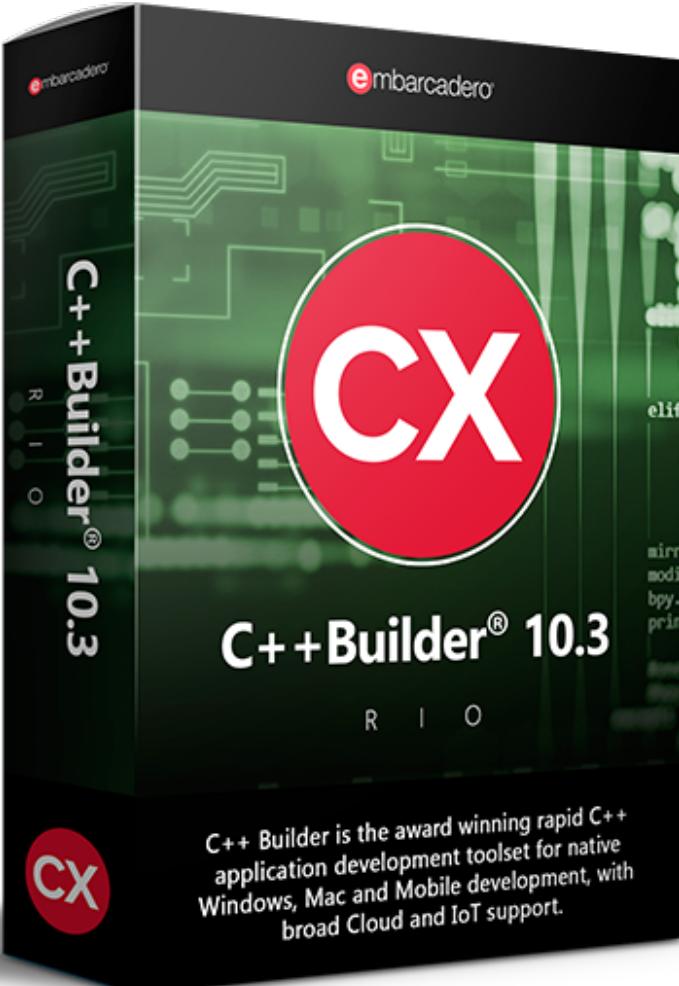
# C++ - Other Enhancements

## Toolchain performance and quality improvements

- Many improvements to CMake support
- Many C++ compiler enhancements, debug info generation, language extensions such as properties and closures, and exception handling
- The RTL has been updated with several missing methods that may be relied on by third party or open source C++ libraries
- General improvements, e.g. a new variadic version of Format(), and two new FreeAndNil signatures
- Support for ARC removal on mobile.



# Questions?



1. Components (VCL / FMX )
2. Common Libraries (RTL)
3. Platform APIs ('Touch the Metal')

# Any Questions?