Embedded System Design Techniques™

Transitioning from C to C++

Class 1: C++ Fundamentals

October 9th, 2017

Jacob Beningo

DesignNews

CEC CONTINUING EDUCATION CENTER

Presented by: Digi-Key ELECTRONICS

Course Overview

Topics:

- **C++ Fundamentals**
- Designing a C++ Application
- Beginning the Transition
- Real-Time C++
- Getting into the Bits and the Bytes

The Lecturer – Jacob Beningo

Jacob Beningo

Principal Consultant

BENINGO EMBEDDED GROUP

Social Media / Contact

- E : jacob@beningo.com
- T : 810-844-1522
- : Jacob_Beningo
- : Beningo Engineering
- : JacobBeningo
- EDN : Embedded Basics
- ARM Connected Community

Consulting

- Advising
- Coaching
- Content
- Consulting
- Training

www.beningo.com

Jacobs CEC Courses

CEC 2013 – 2015

- Fundamentals of Embedded Software (2013)
- Mastering the Software Design Cycle (2014)
- Python for Embedded Systems(2014)
- Software Architecture Design (2014)
- Baremetal C (2015)
- Mastering the ARM Cortex-M Processor (2015)
- Writing Portable and Robust Firmware in C (2015)
- Design Patterns and the Internet (2015)

CEC 2016 2017

- Bootloader Design for MCUs (2016)
- Rapid Prototyping w/ Micro Python (2016)
- Debugging (2016)
- Professional Firmware (2016)
- API's and HAL's February 2017
- Baremetal to RTOS April 2017
- Designing IoT Sensor Nodes July 2017
- From C to C++ October 2017

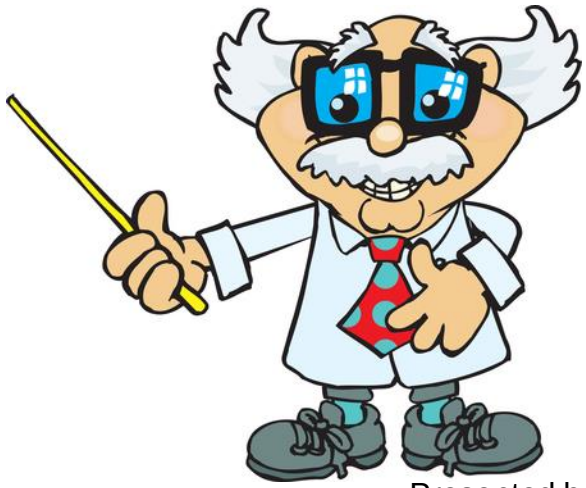
Side Topics 2017

- Real-Time Software using Micro Python
- Embedded Bytes Newsletter

http://bit.ly/1BAHYXm

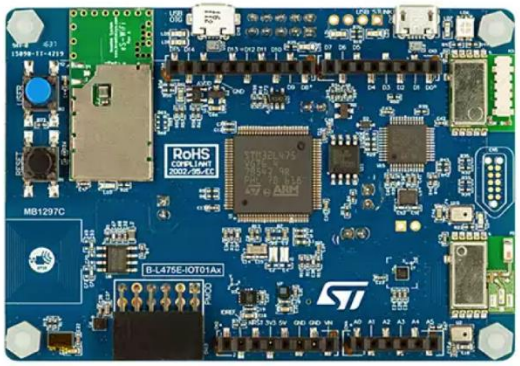
Session Overview

- Introduction to “Embedded C++”
- C versus C++
- Object Oriented Design
- Classes



Presented by:

Course Tools and Software

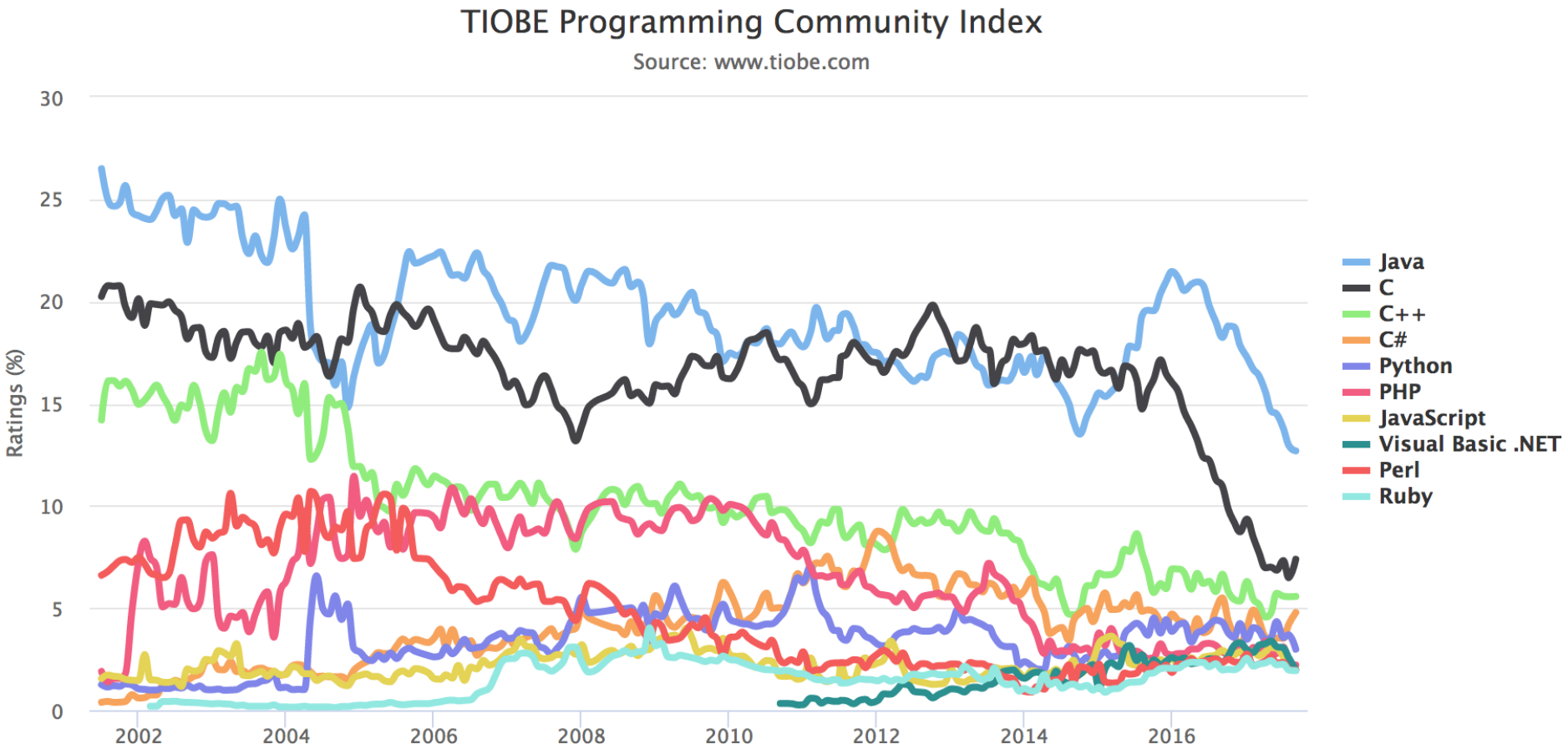


STM32 IoT Discovery Node

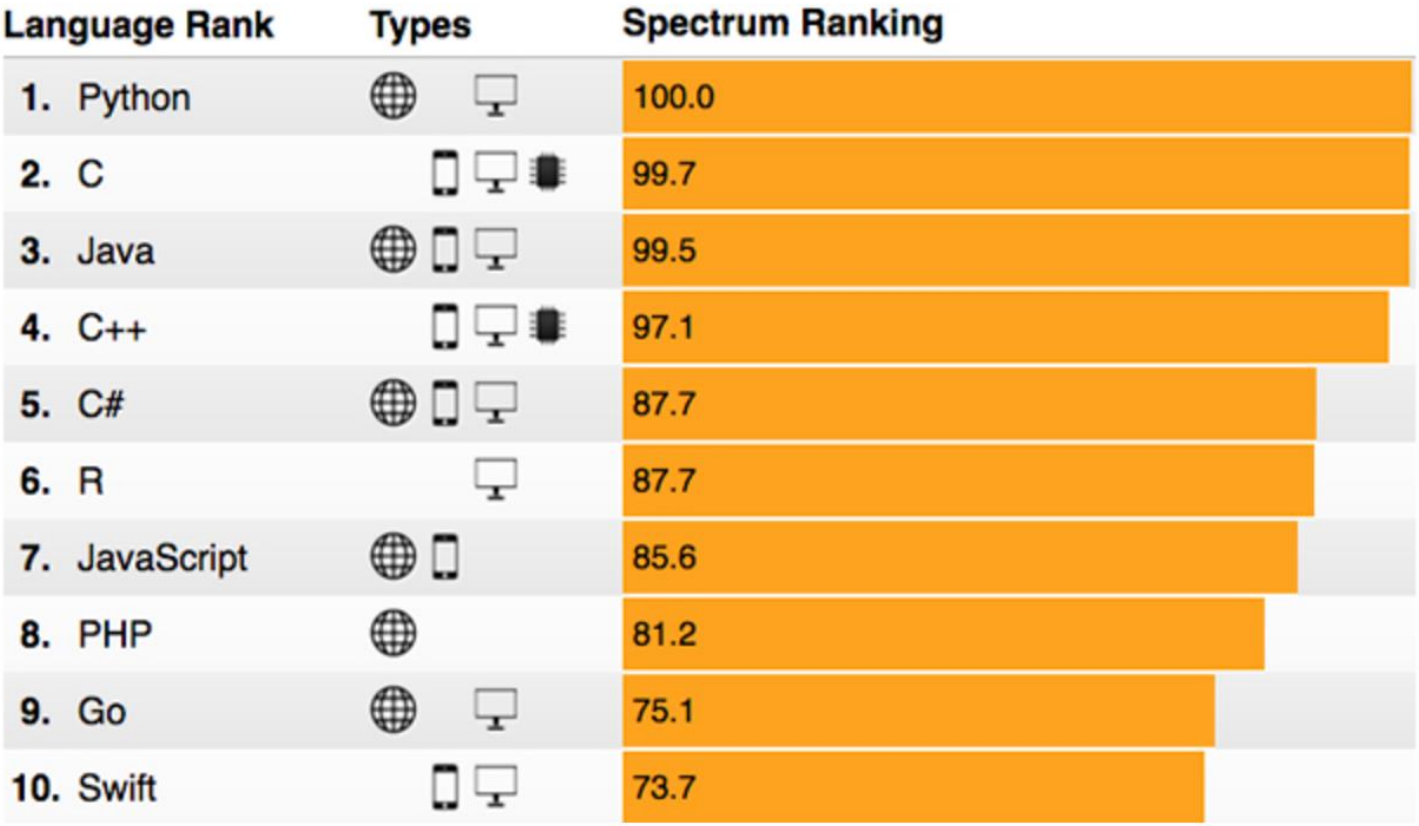
Atollic TrueSTUDIO

GNU G++

Introduction



Introduction



Source: https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

Introduction

- Dennis and Ritchie created C in the 60's
- Bjarne Stroustrup created C++ in 1983*
  - Added data abstraction and object-oriented programming to C
  - Named C++ for ++ operator
  - Standardized in ISO/IEC 14882:1998

C versus C++

- Procedural language – follows a series of well-structured steps and procedures to complete a computational task or program

- Object-Oriented design – A technique for developing software in which the solution is expressed in terms of objects – self-contained entities composed of data and operations on that data

C versus C++

- Procedural
- Low level
- char mystring[] = “hi”;
- No error handling
- Type aliasing

- Object Oriented
- Low and high level
- string mystring = “hi”;
- Try/catch
- Unique types
- type conversion operators
- Operator overloading
- Abstract data types

C to C++ - References

```
void increment(int a)
{
  a++;
}

int value = 2;
int main()
{
  increment(value);
}
```

Value is 2

```
void increment(int &a)
{
  a++;
}

int value = 2;
int main()
{
  increment(value);
}
```

Value is 3

C to C++ - References

C Code

```
void increment(int *a)
{
  (*a)++;
}

int value = 2;
int main()
{
  increment(&value);
}
```

C++ Code

```
void increment(int &a)
{
  a++;
}

int value = 2;
int main()
{
  increment(value);
}
```

C to C++ - Function Overloading

```
void increment(int &a);
void increment(float &a);
int    increment(uint8_t a);
```

Overloading

```
void increment(int &a);
int increment(int a);
float increment(int a);
void increment (int a[]);
```

Invalid!

A simple application

C Application

```
#include <stdio.h>


int main()
{
  printf("Hello World!");
}
```

C++ Application

```
#include <iostream>

using namespace std;

int main()
{
  cout << "Hello World!";
}
```

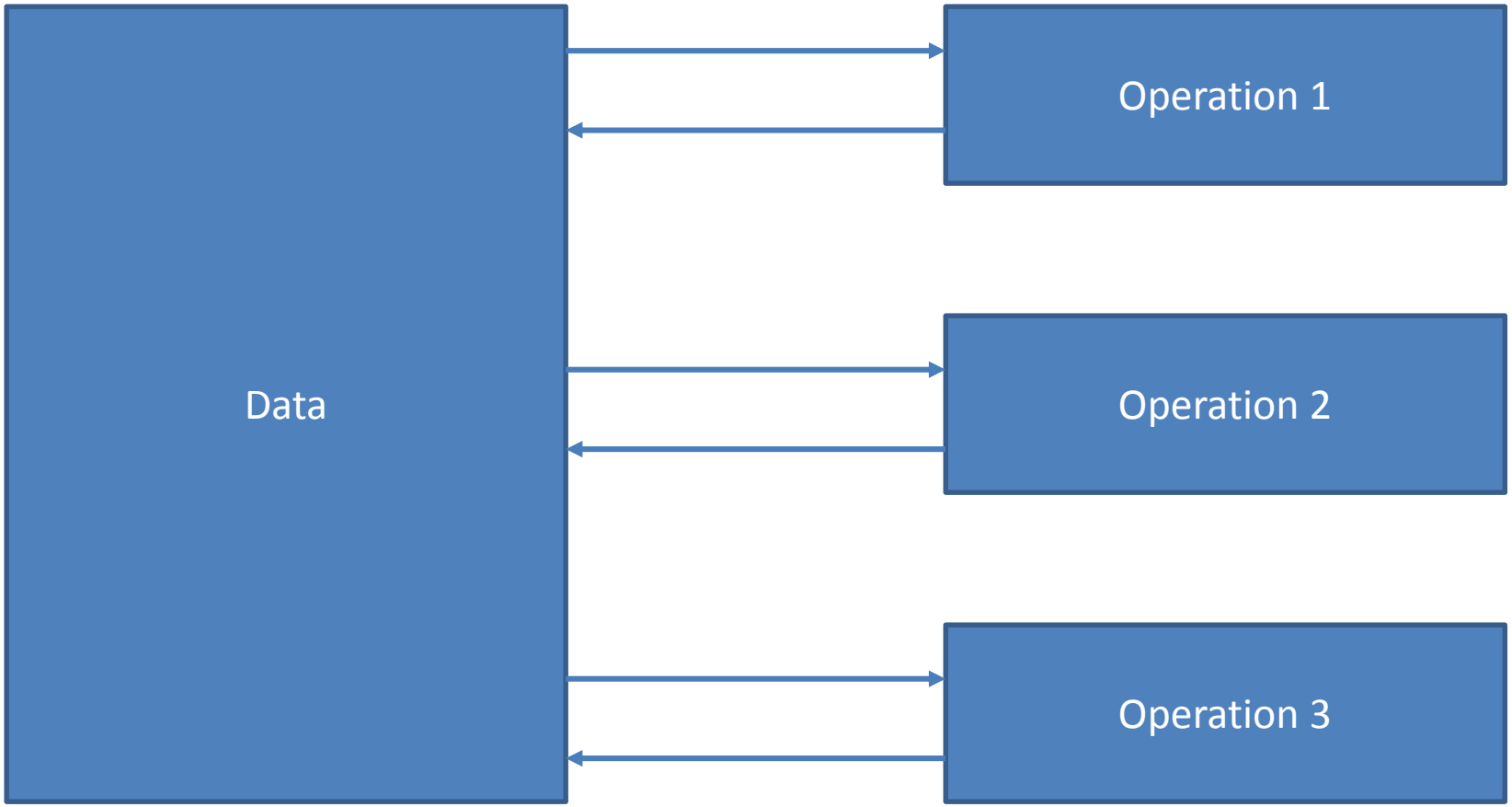
namespaces

```
namespace std
{
   : namespace declarations, i.e. vars, data, etc
}
```

namespace “hides” identifiers within the block

1) Qualified access, std::cout
2) using namespace std

Objects and Classes



Objects and Classes

```
class Time_t
{
public:
        void Set (int, int, int);
        void Increment();
        void PrintTime(); const;
        bool Equal(Time_t) const;
        bool LessThan(Time_t) const;
private:
        int hrs;
        int mins;
        int secs;
};
```

```
Time_t TimeNow;
Time_t TimeStamp;
Time_t TimeStart;
Time_t TimeStop;
Time_t Duration;

TimeNow.Set(0,0,0);
TimeNow.Increment();
```

Objects and Classes

```
#include “timetype.h”
#include <iostream>

void Time_t::Set(int hours, int minutes, int seconds)
{
  hrs = hours;
  mins = minutes;
  secs = seconds;
}

void Time_t::PrintTime() const
{
  cout << hrs << mins << secs;
}
```

Objects and Classes - Constructors

```
class Time_t
{
public:
        void Set (int, int, int);
        void Increment();
        void PrintTime(); const;
        bool Equal(Time_t) const;
        bool LessThan(Time_t) const;
        Time_t (int,int,int);
        Time_t ();
private:
        int hrs;
        int mins;
        int secs;
};
```

```
void Time_t::Time_t(int hours,
                    int minutes,
                    int seconds)
{
  hrs = hours;
  mins = minutes;
  secs = seconds;
}

void Time_t::Time_t()
{
  hrs = 0;
  mins =0;
  secs = 0;
}
```

Additional Resources

- Download Course Material for
  - C/C++ Doxygen Templates
  - Example source code
  - Blog
  - YouTube Videos
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Transitioning from C to C++