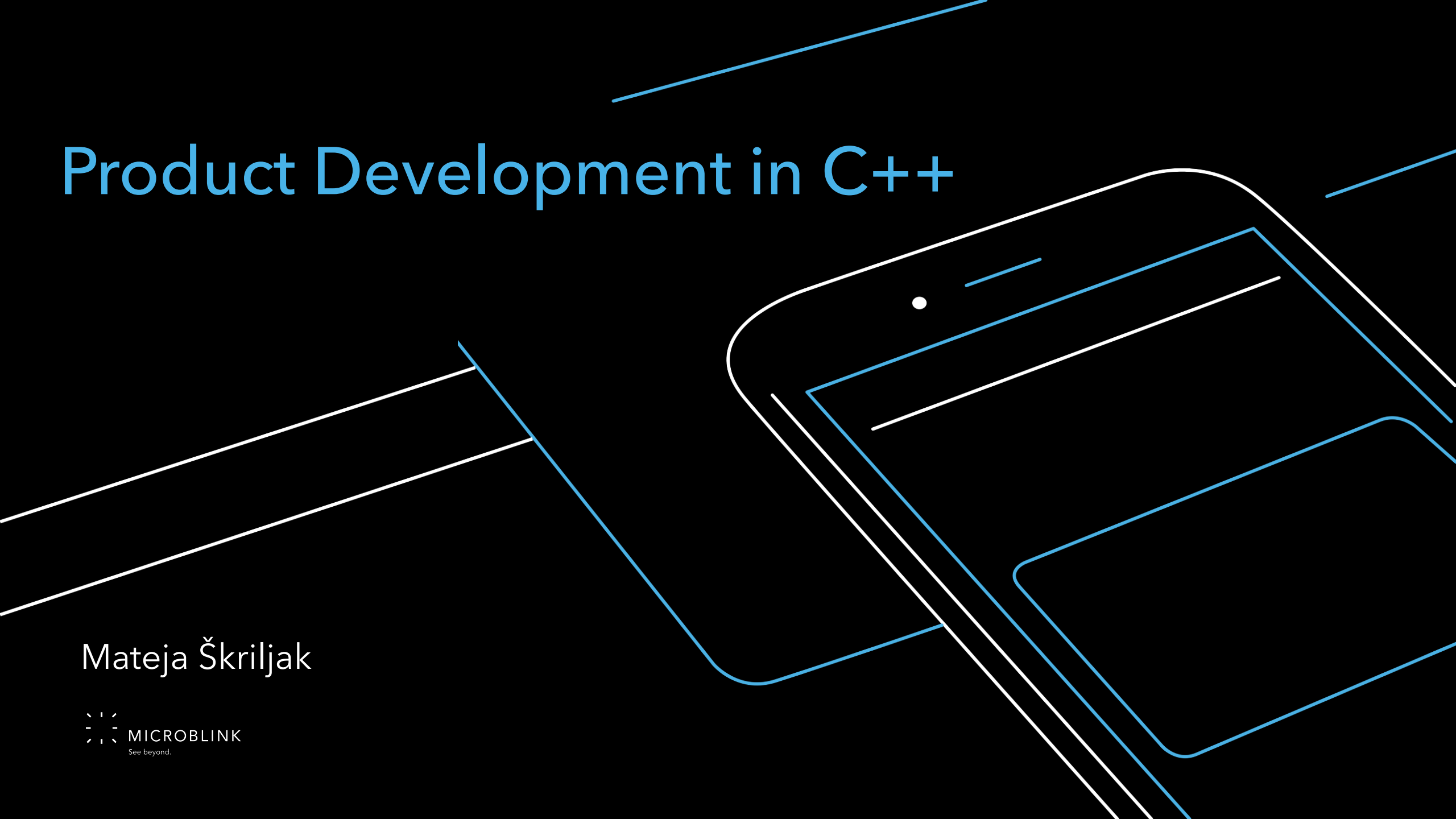
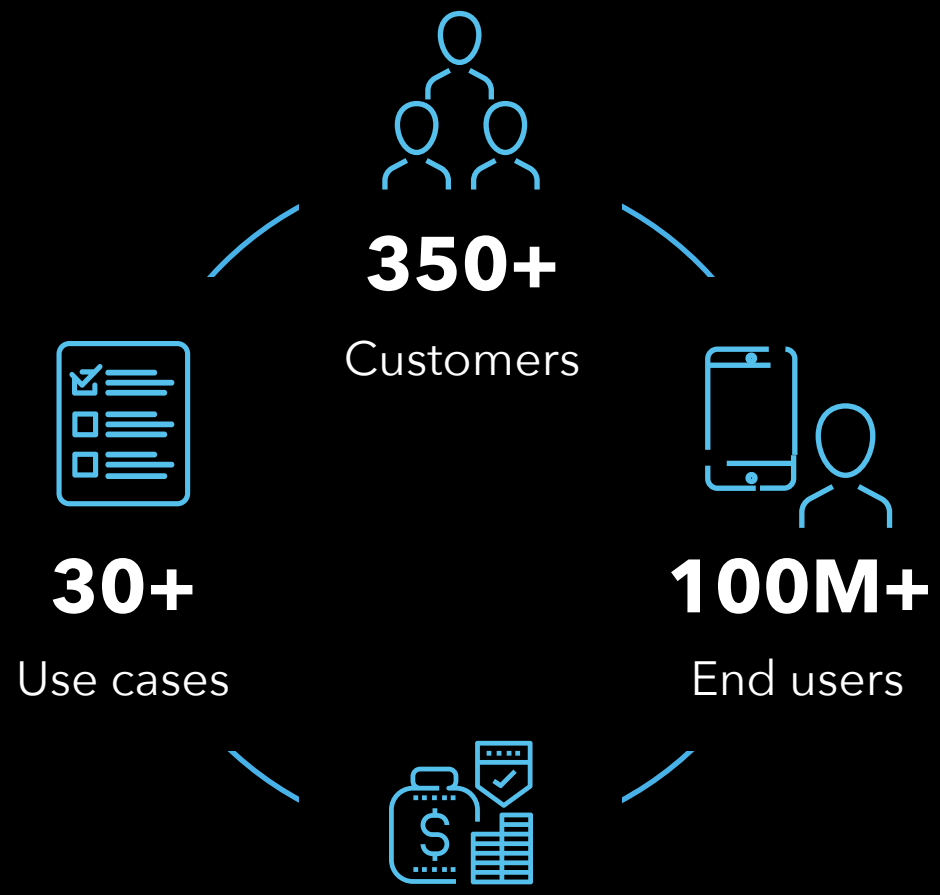
Product Development in C++



Mateja Škriljak

MICROBLINK
See beyond.


350+
Customers
100M+
End users
30+
Use cases
50+
Customers in banking
and telco

BLINK PHOTOPAY

Bills paid briskly.

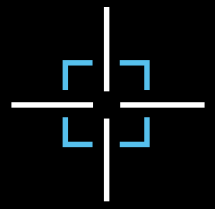

BLINK ID
Identified immediately.
ID CARD
GERMANY
ERIKA
First Name
MUSTERMANN
Surname
T22000129
Document No.
DEUTSCH
Nationality
BERLIN
Place of birth
12/08/1964
Date of birth
31/10/2020
Date of expiry
51147 KÖLN
HEIDERSTRASSE 17
Address
CONFIRM

BLINK RECEIPT

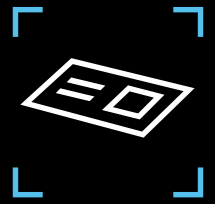
Receipts read rapidly.

Features

Fast recognition



High precision

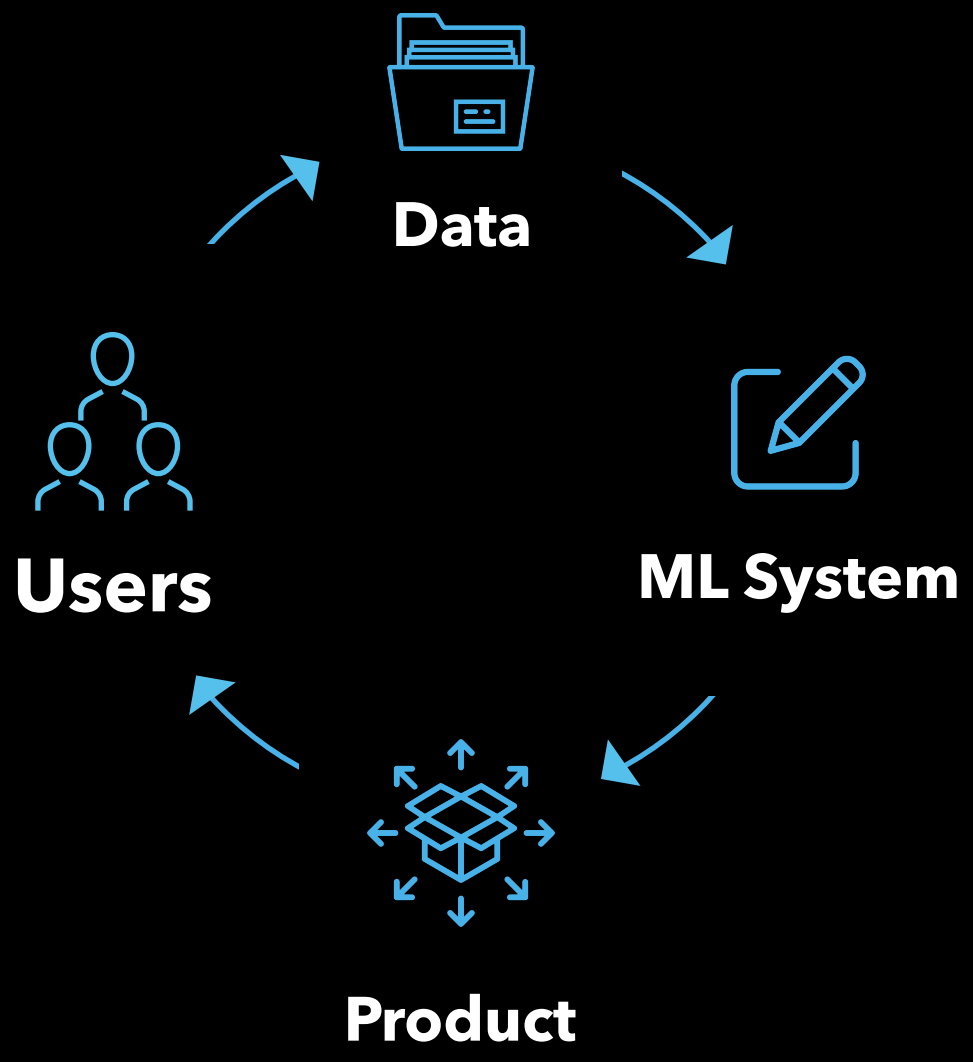


Smart detection

Customisable UI

Works offline

Multichannel


Data
ML System
Product
Users

C++ | Starting Out

- Done lab work in C++? Transitioning from Java, Python, …?
- C background? Useful but not necessary.

- Recommendation: *C++ Primer 5th edition,* Lippman, Lajoie, Moo

C++ | When and Why

- No reason not to use Python or Java?
    - Use Python or Java.

- C++ performance/portability necessary?
    - Don’t just write Python or Java code using C++ syntax.

- Common mistakes: exceptions, streams, dynamic allocation…

C++ | “Modern C++”

- C++11 and after (C++11, C++14, C++17, C++20)
- **auto** keyword - automatic type deduction
- range **for** loop
- **move** semantics
- lambda functions
- **constexpr**
- **concept**…

C++ | Paradigm shift

- “C with classes”
  - **Object oriented** design is no longer enough
- Object hierarchies as their own purpose
  - Bloated code with little benefit
- KISS becomes hard to maintain as problems get more abstract

C++ | Paradigm Shift | Functional Programming Inspired

- First class functions
- Higher order functions
- Immutability - **const**, **constexpr**

```
operator()
```

```
std::function< double ( double, double ) > sum
```

```
auto sum = [] ( double x, double y ) { return x + y; };
```

```
std::transform( v.begin(), v.end(), v.begin(), [] ( auto n ) { return 2 * n; }
```

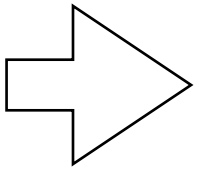
C++ | Paradigm Shift | Policy-Based Design

```
struct Shape
{
      ...
      virtual void draw( Context & ) const = 0;
      ...
};

struct Rectangle : public Shape
{
      ...
      void draw( Context & ) const override { ... }
      ...
};

struct Circle : public Shape
{
      ...
      void draw( Context & ) const override { ... }
      ...
};
```

```
struct WindowWriter
{
      ...
      void drawShape( Shape const * const shape )
      {
            shape->draw( context );
      }
      ...
      Context context;
};
```



```
struct Rectangle
{
      ...
      void draw( Context & ) const { ... }
      ...
};

struct Circle
{
      ...
      void draw( Context & ) const { ... }
      ...
};
```

```
template< typename Drawable >
struct WindowWriter
{
      ...
      void drawObject( Drawable const & obj )
      {
            obj.draw( context );
      }
      ...

      Context context;
};
```

C++ | Policy-Based Design

- "Reverses" the interface-implementation hierarchy
- C++20: **concepts**

- *Modern C++ Design: Generic Programming and Design Patterns Applied*, A. Alexandrescu

https://github.com/dutor/loki

C++ | Modern Principles of Use

- Heap vs. stack allocation
- **const**-correctness
- Compile-time expression evaluation
  - **constexpr**
- Exception safety, smarter error handling
- Template parameter constraints
  - **std::enable_if**, **concept**

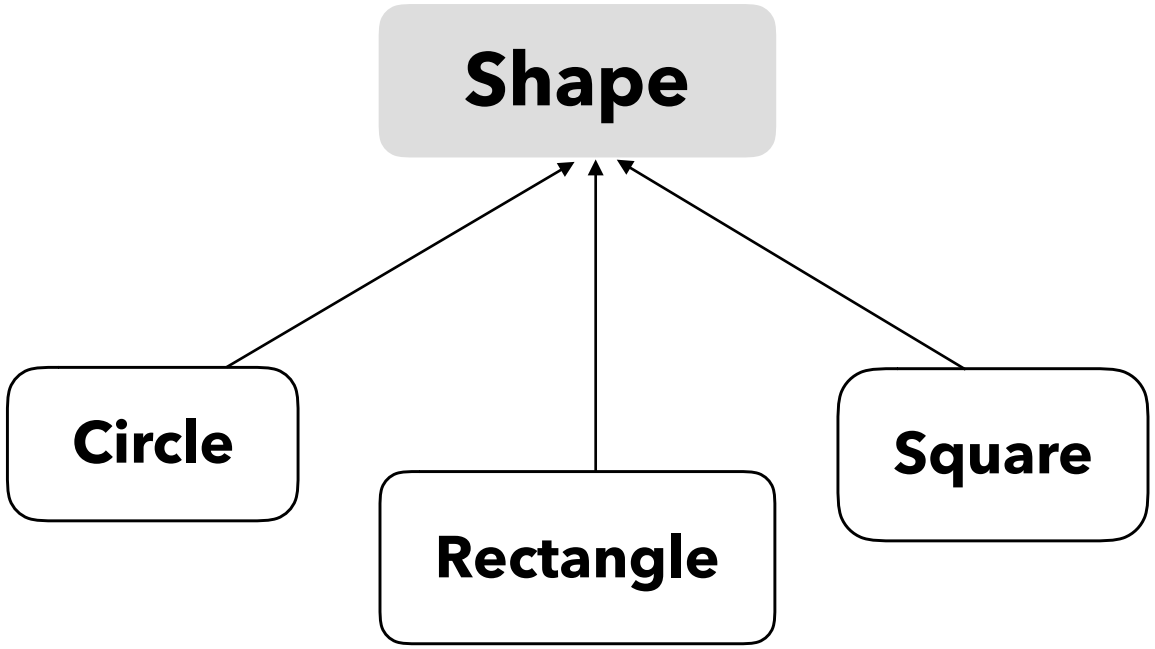
C++ | Example: Traditional Usage

```
class Shape
{
public:
    virtual ~Shape();
    virtual float area() const = 0;
    virtual Shape * clone() const = 0;
};
```

```
class Rectangle : public Shape
{
public:
    Rectangle( float width, float height );

    float area() const;
    Shape * clone() const;

protected:
    float width_;
    float height_;
};
```




```
Shape * shape = new Rectangle( 4.8f, 4.6f );
```

C++ | Example: Modern Usage

```
using Shape = std::variant< Circle, Rectangle, Square >;
```

```
struct Rectangle
{
    float const width;
    float const height;

    Rectangle( float width, float height ) noexcept;
};

float area( Rectangle const & ) noexcept;
```

Circle

Rectangle

Square

```
Shape shape = Rectangle( 4.8f, 4.6f );
```

https://github.com/DoDoENT/modern-cpp-example

C++ | Example: Stream overhead

Binary size (B)

```cpp
#include <iostream>

int main( int /*argc*/, char ** /*argv*/ )
{
    std::cout << "Output stream characters" << std::endl;

    return 0;
}
```

18916

```cpp
#include <cstdio>

int main( int /*argc*/, char ** /*argv*/ )
{
    printf( "Output stream characters\n" );

    return 0;
}
```

8432

C++ | Interesting Examples

- SFINAE

  https://en.wikibooks.org/wiki/More_C%2B%2B_Idioms/SFINAE

- Compile-time regular expression parsing

  https://github.com/hanickadot/compile-time-regular-expressions

- Constexpr game

  https://jguegant.github.io/blogs/tech/meta-crush-saga.html

C++ | Learning Resources

- Algorithmic base

  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/
  https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-851-advanced-data-structures-spring-2012/

- *Effective Modern C++*, S. Meyers
- IsoCpp

  https://isocpp.org/faq

- Herb Sutter's GotW

  https://herbsutter.com/gotw/

- CppCon

  https://www.youtube.com/user/CppCon/playlists

- Jason Turner's Weekly

  https://www.youtube.com/user/lefticus1/videos

- Compiler Explorer

  https://godbolt.org/

C++ | Professional Development | Microblink - Product

- Git

  https://learngitbranching.js.org/

- Jenkins CI
- Conan package manager

  https://docs.conan.io/en/latest/

- UBSAN, ASAN, MSVC STLAssert, Memcheck, ClangTidy
- QA
- **Code Review**
  - **Two stage** approval: junior review + senior review

C++ | Microblink

- mateja.skriljak@microblink.com
- jobs@microblink.com
- microblink.com/careers