

Modern C++ Course



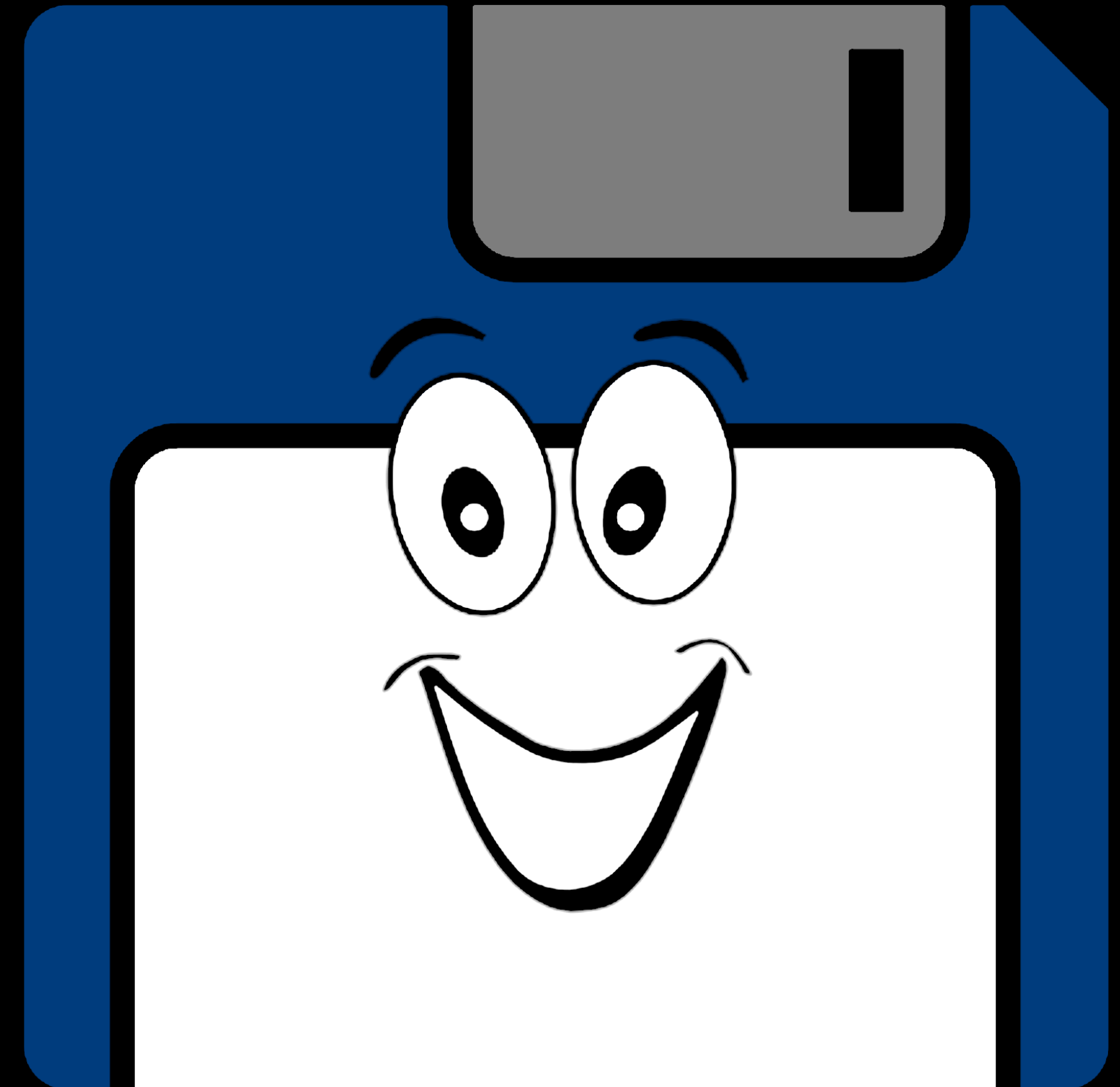
Who am I ?

Gammasoft

Gammasoft aims to make c++ fun again.

About

- Gammasoft is the nickname of Yves Fiumefreddo.
- More than thirty years of passion for high technology especially in development (c++, c#, objective-c, ...).
- Object-oriented programming is more than a mindset.
- more info see my GitHub : <https://github.com/gammasoft71>



Outline

1. Introduction
2. Language Basics
3. Object Oriented Programming (OOP)
4. Core Modern C++
5. Modern C++ Expert
6. Advanced Programming



Outline

1. Introduction
2. Language Basics
3. Object Oriented Programming (OOP)
4. Core Modern C++
5. Modern C++ Expert
6. Advanced Programming



Outline

1. Introduction
2. Language Basics
3. Object Oriented Programming (OOP)
4. Core Modern C++
5. Modern C++ Expert
6. Advanced Programming



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



The classic first application



The classic first application

program.cpp



```
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
}
```



The classic first application

program.cpp



```
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
}
```

CMakeLists.txt



```
cmake_minimum_required(VERSION 3.20)

project(hello_world)
add_executable(${PROJECT_NAME} program.cpp)
```



The classic first application

program.cpp

```
● ● ●  
  
#include <iostream>  
  
int main() {  
    std::cout << "Hello, World!" << std::endl;  
}
```

CMakeLists.txt

```
● ● ●  
  
cmake_minimum_required(VERSION 3.20)  
  
project(hello_world)  
add_executable(${PROJECT_NAME} program.cpp)
```

Output

```
● ● ●  
  
> Hello, World!
```



The classic first application

program.cpp

```
● ● ●  
  
#include <print>  
  
auto main() -> int {  
    std::println("Hello, World!");  
}
```

CMakeLists.txt

```
● ● ●  
  
cmake_minimum_required(VERSION 3.20)  
  
project(hello_world)  
set(CMAKE_CXX_STANDARD 23)  
set(CMAKE_CXX_STANDARD_REQUIRED ON)  
add_executable(${PROJECT_NAME} program.cpp)
```

Output

```
● ● ●  
  
> Hello, World!
```



Main function



```
#include <iostream>

int main() {
    std::cout << "main without arguments" << std::endl;
}
```



```
#include <iostream>

int main(int argc, char* argv[]) {
    std::cout << "main with argc and argv arguments" << std::endl;
}
```



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions




Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Comments



```
// single-line comment
int value = 0;

/*
 * multi-line comment
 */
std::string name();

/// Doxygen comments
/// @brief Adds two specified integers.
/// @param a the first integer to add.
/// @param a the second integer to add.
/// @return The result of the addition.
/// @see https://www.doxygen.nl/manual/commands.html
int add(int a, int b);
```



Basic types



```
bool b = true; // boolean, true or false
```


```
char c = 'a';           // min 8 bit integer  
char cs = -1;           // may be signed  
char cu = '\2';         // or not  
                        // can store an ASCII character
```

```
signed char sc = -3;    // min 8 bit signed integer  
unsigned char uc = 4;   // min 8 bit unsigned integer
```

```
short int si = -5;      // min 16 bit signed integer  
short s = -6;           // int is optional  
unsigned short int usi = 7; // min 16 bit unsigned integer  
unsigned short us = 8;  // int is optional
```



Basic types



```
int i = -9;           // min 16, usually 32 bit
unsigned int ui = 10; // min 16, usually 32 bit

long l = -11l;        // min 32 bit signed integer
long int li = -12l;   // int is optional
unsigned long ul = 13Ul; // min 32 bit unsigned integer
unsigned long int uli = 14Ul; // int is optional

long long ll = -15ll; // min 64 bit signed integer
long long int lli = -16ll; // int is optional
unsigned long long ull = 17ull; // min 64 bit unsigned integer
unsigned long long int ulli = 18ull; // int is optional
```



Basic types



```
float f = 0.19f;           // 32 (1+8+23) bit float
double d = 0.20;           // 64 (1+11+52) bit float
long double ld = 0.21l;    // min 64 bit float

const char* nstr = "native string"; // array of chars ended by \0
std::string str = "string";          // class provided by the STL
```



Fixed width integer type



```
#include <cstdint>
```

```
std::int8_t i8 = -1;    // 8 bit signed integer  
std::uint8_t ui8 = 1;   // 8 bit unsigned integer
```

```
std::int16_t i16 = -2;   // 16 bit signed integer  
std::uint16_t ui16 = 3;  // 16 bit unsigned integer
```

```
std::int32_t i32 = -4;   // 32 bit signed integer  
std::uint32_t ui32 = 5;  // 32 bit unsigned integer
```

```
std::int64_t i64 = -4;   // 64 bit signed integer  
std::uint64_t ui64 = 5;  // 64 bit unsigned integer
```



Integer literals



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- **Inline keyword**
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- **Inline keyword**
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- **Inline keyword**
- Assertions



Language Basics

- Hello World
- Core syntax and types
- Arrays and Pointers
- Scopes / namespaces
- Class and enum types
- References
- Functions
- Operators
- Control structures
- Headers and interfaces
- Auto keyword
- Inline keyword
- Assertions



Outline

1. Introduction
2. Language Basics
3. Object Oriented Programming (OOP)
4. Core Modern C++
5. Modern C++ Expert
6. Advanced Programming



End

