Federal University of Santa Catarina
Real-Time Operating Systems

Introduction to Object-Oriented Programming with C++

Prof. Dr. Giovani Gracioli
giovani@lisha.ufsc.br
http://www.lisha.ufsc.br/Giovani

Objectives

- Introduction to OO with C++
  - Improved C, with more functionalities and with object orientation
- Present concepts of classes and objects
- Explain how to create classes and objects with C++
- Demonstrate how to use classes and objects in C++

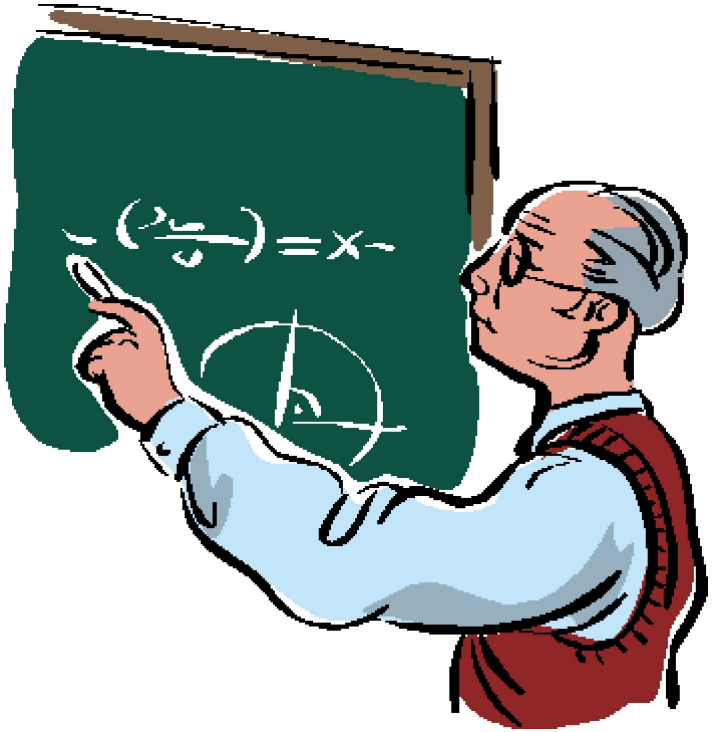
What you need to know to follow

- Good programming skills in C
  - Introduction to Computer Programming
  - Introduction to Data Structures

What you will learn

- Use I/O in C++
- To define a class and use it to create an object
- To define methods and declare attributes of a class
- To call methods to do a task
- To use class constructors to initialize data of an object when it is created

Let's get started



C++ (1)

- C++ improves many resources from the C language and offers the paradigm of Object-Oriented Programming
  - Increases productivity, quality, and re-usability of software
- C++ was developed by Björn Stroustrup, in the Bell Laboratories, and originally was called of “C with classes”
- The name C++ includes the increment operator of C (++) to indicate that C++ is an improved version of C

C++ (2)

- The file names in C have the extension .c (lower case)
- The file names in C++ can have several extensions, like .cpp, .cxx, or .cc
- C++ supports many commands from C
  - while, for, if, switch, do..while, etc..
- And also types
  - int, float, double, struct, enum, pointers, arrays, etc
  - However, the use of libraries is different, example...

C++: I/O example

```
1  // Figura 15.1: fig15_01.cpp
2  // Programa de adição que mostra a soma de dois números.
3  #include <iostream> // permite que o programa realize entrada e saída
4
5  int main()
6  {
7     int number1; // primeiro inteiro a somar
8
9     std::cout << "Digite o primeiro inteiro: "; // pede dados do usuário
10    std::cin >> number1; // lê primeiro inteiro do usuário em number1
11
12    int number2; // segundo inteiro a somar
13    int sum; // soma de number1 e number2
14
15    std::cout << "Digite o segundo inteiro: "; // pede dados do usuário
16    std::cin >> number2; // lê segundo inteiro do usuário em number2
17    sum = number1 + number2; // soma os números; armazena resultado em sum
18    std::cout << "A soma é " << sum << std::endl; // mostra a soma; fim da linha
19 } // fim da função main
```

```
Digite primeiro inteiro: 45
Digite segundo inteiro: 72
A soma é 117
```

Figura 15.1 ■ Programa de adição que exibe a soma de dois números.

Input and Output (1)

- Line 9 uses the standard output stream object – std::cout – and the stream insertion operator, <<, to exhibit the string "Digite o primeiro inteiro: "
- The I/O in C++ are performed by streams of characters
- Thus, when line 9 is executed, it sends the stream "Digite o primeiro inteiro: " to std::cout, which normally is "connected" to the screen

Input and Output (2)

- Line 10 uses the standard input stream object – std::cin – and the operator of stream extraction, >>, to obtain a value from the keyboard
- Cascade output

```
std::cout << "A soma é " << number1 + number2 <<
std::endl;
```

- Or

```
 std::cout << "A soma é " << number1 + number2 <<
"\n";
```

Standard Library of C++

Arquivo de cabeçalho da biblioteca-padrão de C++	Explicação
<cmath>	Contém protótipos para as funções da biblioteca matemática. Esse arquivo de cabeçalho substitui o arquivo de cabeçalho <math.h>.
<cstdlib>	Contém protótipos de função para conversões de números para texto, texto para números, alocação de memória, números aleatórios e outras funções utilitárias diversas. Esse arquivo de cabeçalho substitui o arquivo de cabeçalho <stdlib>.
<ctime>	Contém protótipos de função e tipos para a manipulação de hora e data. Esse arquivo de cabeçalho substitui o arquivo de cabeçalho <time.h>.
<vector>, <list>, <deque>, <queue>, <stack>, <map>, <set>, <bitset>	Esses arquivos de cabeçalho contêm classes que implementam os contêineres da biblioteca-padrão de C++. Os contêineres armazenam dados durante a execução de um programa.
<cctype>	Contém protótipos para as funções que testam caracteres em busca de certas propriedades (por exemplo, se o caractere é um dígito ou um sinal de pontuação) e protótipos para as funções que podem ser usadas para converter letras minúsculas em maiúsculas e vice-versa. Esse arquivo de cabeçalho substitui o arquivo de cabeçalho <ctype.h>.
<cstring>	Contém protótipos para funções de processamento de strings no estilo de C. Esse arquivo de cabeçalho substitui o arquivo de cabeçalho <string.h>.
<typeinfo>	Contém classes para identificação de tipo em tempo de execução (determinando tipos de dados no tempo de execução).
<exception>, <stdexcept>	Esses arquivos de cabeçalho contêm classes que são usadas para tratamento de exceção (discutido no Capítulo 24).
<memory>	Contém classes e funções usadas pela biblioteca-padrão de C++ para alocar memória aos contêineres da biblioteca-padrão de C++. Esse cabeçalho é usado no Capítulo 24.
<fstream>	Contém protótipos para funções que realizam entrada de arquivos no disco e saída para arquivos no disco. Esse arquivo de cabeçalho substitui o arquivo de cabeçalho <fstream.h>.
<string>	Contém a definição da classe string da biblioteca-padrão de C++.
<sstream>	Contém protótipos para as funções que realizam entrada de strings na memória e saída para strings na memória.

Function Overloading (1)

- C++ allows the definition of functions with the same name (they must have different parameters, like types, number, or order)
- This capacity is called function overloading. When a overloaded function is called, the C++ compiler selects the appropriate function by analyzing the number, types, and order of the parameters
- The function overload is used to create functions that do similar tasks with different types

Function Overloading (2)

```
1  // Figura 15.10: fig15_10.cpp
2  // Funções sobrecarregadas.
3  #include <iostream>
4  using namespace std;
5
6  // função square para valores int
7  int square( int x )
8  {
9     cout << "quadrado do int " << x << " é ";
10    return x * x;
11 } // fim da função square com argumento int
12
13 // função square para valores double
14 double square( double y )
15 {
```

Figura 15.10 ■ Funções square sobrecarregadas. (Parte 1 de 2.)

Function Overloading (3)

```
16    cout << "quadrado do double " << y << " é ";
17    return y * y;
18 } // fim da função square com argumento double
19
20 int main()
21 {
22    cout << square( 7 ); // chama versão int
23    cout << endl;
24    cout << square( 7.5 ); // chama versão double
25    cout << endl;
26 } // fim do main
```

```
quadrado do int 7 é 49
quadrado do double 7.5 é 56.25
```

Figura 15.10 ■ Funções square sobrecarregadas. (Parte 2 de 2.)

Exercise

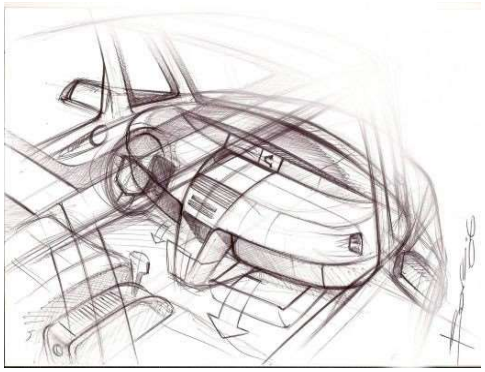
- Write a C++ program that contains different versions of a function, called add, which is able to perform the addition of two integers, of one integer and one float, and three integers. In the main function, call the three function versions to show how it works

Introduction to Classes and Objects

- Classes and objects are two fundamental concepts of any object-oriented programming language
- Normally, your C++ programs will consist of a main function and one or more classes, each one containing member functions (methods) and data members (attributes)

Classes, objects, methods, and attributes (1)

- Analogy: suppose you want to drive a car faster stepping on the throttle
  - First of all, the car must be designed and built
  - The pedal “hides” the complex mechanisms that make the car move fast, the wheel “hides” the mechanisms that make the car turn
  - This allows people that do not know the internals of a car to drive it easily, just using pedals, breaking, wheel, and other simple “interfaces”



Classes, objects, methods, and attributes (2)

- In C++, classes are used to house a function, as drawings in engineering are used to house the project of the throttle pedal
- The function of a class is called member function or method
  - They do the tasks of classes
- As you cannot drive a draw, you cannot "drive" a class either
  - It is necessary to create an object
  - Many objects can be created from the same class, as many cars are built from the same draw

Classes, objects, methods, and attributes (3)

- When a person steps on the throttle, a message is sent to the system that controls the car speed
- Similarly, messages are sent to objects (methods call)
  - Request of a service of an object

Classes, objects, methods, and attributes (4)

- In addition to the functionalities that a car offers, it also has several attributes, like color, number of doors, KM, etc
  - These attributes are also represented in the car's project
  - Each car has its own attributes
- Similarly, each object has attributes that are loaded when it is created
  - These attributes or data members are specified as part of the object

Example: how to create a class

```
1  // Fig. 16.1: fig16_01.cpp
2  // Define a classe GradeBook com uma função-membro displayMessage,
3  // cria um objeto GradeBook e chama sua função displayMessage.
4  #include <iostream>
5  using namespace std;
6
7  // Definição da classe GradeBook
8  class GradeBook
9  {
10 public:
11    // função que mostra uma mensagem de boas-vindas ao usuário de GradeBook
12    void displayMessage()
13    {
14       cout << "Bem-vindo ao Grade Book!" << endl;
15    } // fim da função displayMessage
16 }; // fim da classe GradeBook
17
18 // função main inicia a execução do programa
19 int main()
20 {
21    GradeBook myGradeBook; // cria um objeto GradeBook chamado myGradeBook
22    myGradeBook.displayMessage(); // chama função displayMessage do objeto
23 } // fim do main
```

```
Bem-vindo ao Grade Book!
```

Figura 16.1 ■ Define a classe GradeBook com uma função-membro displayMessage, cria um objeto GradeBook e chama sua função displayMessage.

Defining a class with a Method

- Normally, you cannot call a class method until creating an object of this class (line 21)
  - The variable type is GradeBook
  - It is a user-defined type
  - Each created class is a new type that can be used to create objects
- In line 22, the method displayMessage is called using the object myGradeBook, followed by point operator (.)

Defining a method with a parameter (1)

```
1  // Fig. 16.3: fig16_03.cpp
2  // Define classe GradeBook com função-membro que usa um parâmetro;
3  // Cria um objeto GradeBook e chama sua função displayMessage.
4  #include <iostream>
5  #include <string> // programa usa classe de string padrão de C++
6  using namespace std;
7
8  // Definição da classe GradeBook
9  class GradeBook
10 {
11 public:
12    // função que mostra mensagem de boas-vindas ao usuário do GradeBook
13    void displayMessage( string courseName )
14    {
15       cout << "Bem-vindo ao grade book para\n" << courseName << "!"
16          << endl;
17    } // fim da função displayMessage
18 }; // fim da classe GradeBook
19
20 // função main inicia a execução do programa
21 int main()
22 {
```

Defining a method with a parameter (2)

```
23     string nameOfCourse; // string de caracteres para armazenar o nome do curso
24     GradeBook myGradeBook; // cria um objeto GradeBook chamado myGradeBook
25
26     // pede e insere nome do curso
27     cout << "Favor informar o nome do curso:" << endl;
28     getline( cin, nameOfCourse ); // lê um nome de curso com espaços
29     cout << endl; // mostra uma linha em branco
30
31     // chama função displayMessage de myGradeBook
32     // e passa nameOfCourse como argumento
33     myGradeBook.displayMessage( nameOfCourse );
34  } // fim do main
```

```
Favor informar o nome do curso:
CS101 Introdução à programação C++

Bem-vindo ao grade book para
CS101 Introdução à programação C++!
```

Figura 16.3 ■ Define classe GradeBook com uma função-membro que usa um parâmetro, cria um objeto GradeBook e chama sua função displayMessage.

Attributes, set and get methods (1)

- Variables declared inside a function are known as local variables
  - Must be declared before their usage, are not accessed outside the function, and are "destroyed" when the function ends
- Attributes of an object are represented as variables in a class definition
  - They are declared inside the class definition, but outside the methods
  - Methods manipulate the attributes of an object
- Each object has its own copy of the attributes in memory

Attributes, set and get methods (2)

```
1  // Fig. 16.5: fig16_05.cpp
2  // Define classe GradeBook que contém um dado-membro courseName
3  // e funções-membro para definir e obter seu valor;
4  // Cria e manipula um objeto GradeBook com essas funções.
5  #include <iostream>
6  #include <string> // programa usa classe string-padrão da C++
7  using namespace std;
8
9  // Definição da classe GradeBook
10 class GradeBook
11 {
12 public:
13    // função que define o nome do curso
14    void setCourseName( string name )
15    {
16       courseName = name; // armazena o nome do curso no objeto
17    } // fim da função setCourseName
18
19    // função que obtém o nome do curso
20    string getCourseName()
21    {
22       return courseName; // retorna o courseName do objeto
23    } // fim da função getCourseName
24
25    // função que mostra uma mensagem de boas-vindas
26    void displayMessage()
27    {
28       // essa instrução chama getCourseName para obter o
29       // nome do curso que esse GradeBook representa
30       cout << "Bem-vindo ao grade book para\n" << getCourseName() << "!"
```

Figura 16.5 ■ Definições e teste da classe GradeBook com um dado-membro e funções *set* e *get*. (Parte 1 de 2.)

Attributes, set and get methods (3)

```
31          << endl;
32       } // fim da função displayMessage
33    private:
34       string courseName; // nome do curso para esse GradeBook
35    }; // fim da classe GradeBook
36
37    // função main inicia a execução do programa
38    int main()
39    {
40       string nameOfCourse; // string de caracteres para armazenar o nome do curso
41       GradeBook myGradeBook; // cria um objeto GradeBook chamado myGradeBook
42
43       // exibe valor inicial de courseName
44       cout << "Nome inicial do curso é: " << myGradeBook.getCourseName()
45          << endl;
46
47       // solicita, insere e define nome do curso
48       cout << "\nFavor digitar o nome do curso:" << endl;
49       getline( cin, nameOfCourse ); // lê um nome de curso com espaços
50       myGradeBook.setCourseName( nameOfCourse ); // define o nome do curso
51
52       cout << endl; // gera uma linha em branco
53       myGradeBook.displayMessage(); // exibe mensagem com novo nome do curso
54    } // fim do main
```

```
Nome inicial do curso é:

Favor digitar o nome do curso:
CS101 Introdução à programação C++

Bem-vindo ao grade book para
CS101 Introdução à programação C++!
```

Figura 16.5 ■ Definições e teste da classe GradeBook com um dado-membro e funções *set* e *get*. (Parte 2 de 2.)

Initializing objects with constructors (1)

- Each class must offer a constructor that can be used to initialize an object of that class when it is created
- A constructor is a special method that must be defined with the same name of the class (thus the compiler knows it is a constructor)
- Constructors cannot return values

Initializing objects with constructors (2)

- C++ requires a call to a constructor for each created object, which helps in keeping the object initialization before its usage
- The call occurs implicitly when the object is created
- If a class does not explicitly include a constructor, the compiler will include a standard constructor – a constructor with no parameters

Initializing objects with constructors (3)

```
4  // quando cada objeto GradeBook for criado.
5  #include <iostream>
6  #include <string> // programa usa classe de string C++ padrão
7  using namespace std;
8
9  // Definição de classe GradeBook
10 class GradeBook
11 {
12 public:
13    // construtor inicializa courseName com string fornecida como argumento
14    GradeBook( string name )
15    {
16       setCourseName( name ); // chama função set para inicializar courseName
17    } // fim do construtor GradeBook
18
19    // função para definir o nome do curso
20    void setCourseName( string name )
21    {
22       courseName = name; // armazena o nome do curso no objeto
23    } // fim da função setCourseName
24
25    // função para obter o nome do curso
26    string getCourseName()
27    {
28       return courseName; // retorna courseName do objeto
29    } // fim da função getCourseName
30
31    // exibe mensagem de boas-vindas para o usuário do GradeBook
32    void displayMessage()
```

Initializing objects with constructors (4)

```
33    {
34       // chama getCourseName para obter o courseName
35       cout << "Bem-vindo ao grade book para\n" << getCourseName()
36          << "!" << endl;
37    } // fim da função displayMessage
38 private:
39    string courseName; // nome do curso para esse GradeBook
40 }; // fim da classe GradeBook
41
42 // função main inicia a execução do programa
43 int main()
44 {
45    // cria dois objetos GradeBook
46    GradeBook gradeBook1( "CS101 Introdução à programação C++" );
47    GradeBook gradeBook2( "CS102 Estrutura de dados em C++" );
48
49    // exibe valor inicial de courseName para cada GradeBook
50    cout << "gradeBook1 criado para o curso: " << gradeBook1.getCourseName()
51       << "\ngradeBook2 criado para o curso: " << gradeBook2.getCourseName()
52       << endl;
53 } // fim do main
```

```
gradeBook1 criado para o curso: CS101 Introdução à programação C++
gradeBook2 criado para o curso: CS102 Estruturas de dados em C++
```

Figura 16.7 ■ Instanciando vários objetos da classe GradeBook usando o construtor de GradeBook para especificar o nome do curso quando cada objeto GradeBook for criado. (Parte 2 de 2.)

Review

- A class is formed by a set of methods and attributes that carry out a specific task
- Each method manipulates the attributes of a class
- Each object has its own attributes
  - Accessed by the methods set and get
- Constructors are called to initialize objects before their usage

Tasks

- Read chapters 2, 3, 4, 5, and 6 of the text book (C++ how to program 8th edition)
- Lists of exercises are at moodle

References

- Paul Deitel e Harvey Deitel, C++: como programar, 5a edição, Ed. Prentice Hall Brasil, 2006.
- Paul Deitel e Harvey Deitel, C++: how to program, 8th edition, Ed. Prentice Hall, 2012.