# ESTRUTURAS DE DADOS

## Classes

# Roteiro

- **Noções Básicas**
- **A classe Time**

# Noções Básicas

**Em disciplinas anteriores, você já foi introduzido aos conceitos de Programação Orientada a Objetos (POO).**

**Não entraremos em muitos detalhes teóricos, apenas queremos conhecer a sintaxe de C++.**

**Novos conceitos serão introduzidos sempre que necessário.**

A unidade básica da POO é a **classe**, que encapsula atributos estáticos e comportamento dinâmicos em uma caixa.

A **classe** é um modelo usado para criar **objetos**, também chamados de **instâncias**.

A comunicação com os objetos é feita pelo uso da **interface pública** do objeto.

A complexidade envolvida na realização de uma tarefa fica **escondida** dentro da classe.

**Uma vantagem do paradigma é o isolamento, quando alterações não afetam todo o sistema.**

**Esse isolamento facilita a adição de novas funcionalidades e correção de problemas.**

**Como os detalhes de implementação ficam escondidos dentro da classe, podemos gerar implementações diferentes facilmente.**

**Em Estruturas de Dados, isso permitirá separar a visão lógica, da visão de implementação e da visão de aplicação.**

A visão lógica de uma classe será sempre criada em um arquivo de extensão .h.

Nesse arquivo, definiremos os membros públicos e privados de uma classe.

A implementação da classe será sempre feita em um arquivo de extensão .cpp.

O arquivo .cpp deverá sempre importar o arquivo .h com a diretiva include.

**É comum tentar incluir uma definição de classes mais de uma vez. Nesse caso, utilizamos algumas diretivas que impedem que isso aconteça:**

- **#ifndef: se não definido**
  - **Pule este código se já tiver sido incluído.**
- **#define**
  - **Define um nome para evitar dupla inclusão.**
- **#endif**

```
#ifndef TIME_H   // Inclua esse bloco apenas se TIME_H não está definido
#define TIME_H   // Na primeira inclusão, defina TIME_H para que este
                 // bloco não seja incluído mais de uma vez.

// ...

#endif  // Final do bloco "#ifndef"
```

# A Classe Time

**Definiremos uma classe chamada Time que modela uma instância de tempo: hora, minuto e segundo.**

- **Três atributos (hora, minuto e segundo).**
- **Um construtor público que inicializa os atributos.**
- **Métodos "get" e "set" para gerenciar os atributos.**
- **Método público "print" para imprimir esta instância de tempo no formato hh:mm:ss.**
- **Um método público que adiciona um segundo.**

**Arquivo: time.h**

```
class Time {
 private:  // Seção Privada
  // Membros privados
  int hour;     // 0 - 23
  int minute;   // 0 - 59
  int second;   // 0 - 59
 public:   // Seção Pública
  Time(int hour = 0, int minute = 0, int second = 0);
  int getHour() const;
  void setHour(int hour);
  int getMinute() const;
  void setMinute(int minute);
  int getSecond() const;
  void setSecond(int second);
  void print() const;
  void nextSecond();
};
```

**No arquivo time.cpp, devemos incluir o arquivo time.h.**

```
/* Implementação da classe Time */
#include <iostream>  // Para usar a função std::cout.
#include "time.h"    // Para visualizar a classe Time.

using namespace std; // Para escrever cout ao invés de
                     // std::cout.
```

**O próximo passo é incluir a implementação para todos os métodos. Para tanto, usamos o operador :: de resolução de escopo.**

```
int Time::getHour() const {
  return hour;
}
```

**Arquivo: time.cpp**

```
/*
    Getters
 */
int Time::getHour() const {
  return hour;
}

int Time::getMinute() const {
  return minute;
}

int Time::getSecond() const {
  return second;
}
```

**Arquivo: time.cpp**

```
/*
  Setters
 */
void Time::setHour(int hour) {
  this->hour = hour;
}

void Time::setMinute(int minute) {
  this->minute = minute;
}

void Time::setSecond(int second){
  this->second = second;
}
```

```cpp
void Time::print() const {
  cout << hour << ":" << minute << ":" << second << endl;
}

void Time::nextSecond() {
  second += 1;
  if (second >= 60) {
    second = 0;
    minute += 1;
  }
  if (minute >= 60) {
    minute = 0;
    hour += 1;
  }
  if (hour >= 24) {
    hour = 0;
  }
}
```

**Arquivo: time.cpp**

**Agora que temos a classe pronta, podemos utilizá-la em algum outro ponto do código.**

```
Time t1(23, 59, 59);
t1.print();         // 23:59:59
t1.setHour(12);
t1.setMinute(30);
t1.setSecond(15);

t1.print();         // 12:30:15
cout << "Hour:     " << t1.getHour()   << endl;
cout << "Minute:   " << t1.getMinute() << endl;
cout << "Second:   " << t1.getSecond() << endl;
```

**Arquivo: time_main.cpp**

## Outros exemplos de utilização:

```
Time t2(12);
t2.print();  // 12:00:00

Time t3(23, 59, 58);
t3.print();  // 23:59:58
t3.nextSecond();
t3.print();  // 23:59:59
t3.nextSecond();
t3.print();  // 00:00:00
```

**Arquivo: time_main.cpp**

## Para compilar o código completo, fazemos:

```
$ g++ time_main.cpp time.cpp -o time
$ ./time
```