Engenharia Eletrônica

Orientação a Objetos
-
Programação em C++

1º Slides – B : Introdução à OO/C++

Passando à Pratica – Estruturas, Classes e Objetos

Prof. Jean Marcelo SIMÃO DAINF/UTFPR

Passando à Prática

Nosso primeiro exemplo irá transformar um programa C

em um programa C++

Exercício – Criar um programa em C com os seguintes requisitos:

- Crie um programa em C que permita calcular a idade de duas pessoas.
- Para tal, crie uma *struct* Pessoa, com 'campos' dia, mês, ano e idade.
- Crie duas variáveis de Pessoa, chamadas *Einstein* e *Newton*, na função principal (a função *main*).
- Ainda na função *main*, faça os campos dia, mês e ano de Einstein e Newton receberem valores apropriados.
  - Obs.: Einstein nasceu 14/03/1879 em e Newton em 04/01/1643.
- O cálculo da idade se dará comparando o ano de nascimento com o ano atual.
- Este cálculo se dará em uma função.
- Esta função recebe como parâmetros de entrada: (a) uma variável Pessoa; e (b) uma constante inteira relativa ao ano atual.
- Essa função devolve a idade calculada.
- Faça com que o campo idade de Einstein e o campo idade de Newton recebam valores calculados a partir de chamadas apropriadas do função citada.

Exemplo em C – Versão 1

```c
#include <stdio.h>

struct Pessoa
{
   int dia;
   int mes;
   int ano;
   int idade;
};
```

```c
int Calc_Idade ( struct Pessoa p, int ano )
{
    int idd = ano - p.ano;

    return idd;
}
```

```c
int main()
{
  struct Pessoa Einstein, Newton;

  Einstein.dia    = 14;
  Einstein.mes  = 3;
  Einstein.ano  = 1879;

  Newton.dia    = 4;
  Newton.mes  = 1;
  Newton.ano   = 1643;

  Einstein.idade = Calc_Idade ( Einstein, 2009 );
  Newton.idade  = Calc_Idade ( Newton, 2009 );

  printf("A idade de Einstein seria %d \n", Einstein.idade);
  printf("A idade de Newton seria %d  \n", Newton.idade);

  getchar();
  return 0;
}
```

Exercício – Mudar o programa segundos os novos requisitos:

- O cálculo da idade se dará comparando a data de nascimento com a data atual.
- Este cálculo se dará em uma função.
- Esta função recebe como parâmetros de entrada: (a) uma variável Pessoa; e (b) três constantes inteiras relativas ao dia, mês e ano atual.
- Essa função devolve a idade calculada.

Exemplo em C – Versão 2

```c
#include <stdio.h>

struct Pessoa
{
   int dia;
   int mes;
   int ano;
   int idade;
};
```

```c
int Calc_Idade(struct Pessoa p, int dia, int mes, int ano)
{
   int idd = ano - p.ano;

   if (p.mes > mes)
   {
      idd = idd - 1;
   }
   else
   {
        if (p.mes == mes)
        {
            if (p.dia > dia)
            {
               idd = idd - 1;
            }
        }
   }
   return idd;
}
```

```c
int main()
{
   struct Pessoa Einstein, Newton;

   Einstein.dia    = 14;
   Einstein.mes  = 3;
   Einstein.ano  = 1879;

   Newton.dia    = 4;
   Newton.mes  = 1;
   Newton.ano   = 1643;

   Einstein.idade = Calc_Idade ( Einstein, 28, 8, 2009);
   Newton.idade  = Calc_Idade ( Newton, 28, 8, 2009);

   printf("A idade de Einstein seria %d \n", Einstein.idade);
   printf("A idade de Newton seria %d  \n", Newton.idade);

   getchar();   // serve para esperar caracter – segurar tela
   return 0;
}
```

Exemplo em C – Versão 2

```c
#include <stdio.h>

struct Pessoa
{
   int dia;
   int mes;
   int ano;
   int idade;
};
```

```c
int Calc_Idade(struct Pessoa p, int dia, int mes, int ano)
{
   p.idade = ano - p.ano;

   if (p.mes > mes)
   {
        p.idade = p.idade - 1;
   }
   else
   {
        if (p.mes == mes)
        {
            if (p.dia > dia)
            {
                p.idade = p.idade  - 1;
            }
        }
   }
   return p.idade;
}
```

```c
int main()
{
   struct Pessoa Einstein, Newton;

   Einstein.dia    = 14;
   Einstein.mes  = 3;
   Einstein.ano  = 1879;
   Einstein.idade = -1;

   Newton.dia    = 4;
   Newton.mes  = 1;
   Newton.ano   = 1643;
   Newton.idade = -1;

   Einstein.idade = Calc_Idade ( Einstein, 28, 8, 2009);
   Newton.idade  = Calc_Idade ( Newton, 28, 8, 2009);

   printf("A idade de Einstein seria %d \n", Einstein.idade);
   printf("A idade de Newton seria %d  \n", Newton.idade);

   getchar();
   return 0;
}
```

Exemplo em C – Versão 2’

```c
#include <stdio.h>

struct Pessoa
{
   int dia;
   int mes;
   int ano;
   int idade;
};
```

```c
void Calc_Idade(struct Pessoa p, int dia, int mes, int ano)
{
   p.idade = ano - p.ano;

   if (p.mes > mes)
   {
        p.idade = p.idade - 1;
   }
   else
   {
        if (p.mes == mes)
        {
            if (p.dia > dia)
            {
                p.idade = p.idade  - 1;
            }
        }
   }

}
```

```c
int main()
{
   struct Pessoa Einstein, Newton;

   Einstein.dia    = 14;
   Einstein.mes  = 3;
   Einstein.ano  = 1879;
   Einstein.idade = -1;

   Newton.dia    = 4;
   Newton.mes  = 1;
   Newton.ano   = 1643;
   Newton.idade = -1;

   Calc_Idade ( Einstein, 28, 8, 2009);
   Calc_Idade ( Newton, 28, 8, 2009);

   printf("A idade de Einstein seria %d \n", Einstein.idade);
   printf("A idade de Newton seria %d  \n", Newton.idade);

   getchar();
   return 0;
}
```

Funciona?

Exemplo em C – Versão 3

```c
#include <stdio.h>

struct Pessoa
{
   int dia;
   int mes;
   int ano;
   int idade;
};
```

```c
void Calc_Idade (struct Pessoa *p, int dia, int mes, int ano)
{
   p->idade = ano – p->ano;

   if ( p->mes > mes )
   {
      p->idade = p->idade - 1;
   }
   else
   {
       if ( p->mes == mes )
       {
           if ( p->dia > dia )
           {
              p->idade = p->idade - 1;
           }
       }
   }
}
```

```c
int main()
{
   struct Pessoa Einstein, Newton;

   Einstein.dia    = 14;
   Einstein.mes  = 3;
   Einstein.ano  = 1879;

   Newton.dia    = 4;
   Newton.mes  = 1;
   Newton.ano   = 1643;

   Calc_Idade ( &Einstein, 24, 8, 2009);
   Calc_Idade ( &Newton, 24, 8, 2009);

   printf("A idade de Einstein seria %d \n", Einstein.idade);
   printf("A idade de Newton seria %d  \n", Newton.idade);

   getchar();
   return 0;
}
```

Exemplo em C – Versão 3 B

```c
#include <stdio.h>

struct Pessoa
{
   int dia;
   int mes;
   int ano;
   int idade;
};
```

```c
void Calc_Idade (struct Pessoa *p, int dia, int mes, int ano)
{
   (*p).idade = ano – (*p).ano;

   if ( (*p).mes > mes )
   {
       (*p). idade = (*p).idade - 1;
   }
   else
   {
        if ( (*p).mes == mes )
        {
            if ( (*p).dia > dia )
            {
                (*p). idade = (*p).idade - 1;
            }
        }
   }
}
```

```c
int main()
{
   struct Pessoa Einstein, Newton;

   Einstein.dia    = 14;
   Einstein.mes  = 3;
   Einstein.ano  = 1879;

   Newton.dia    = 4;
   Newton.mes  = 1;
   Newton.ano   = 1643;

   Calc_Idade ( &Einstein, 24, 8, 2009);
   Calc_Idade ( &Newton, 24, 8, 2009);

   printf("A idade de Einstein seria %d \n", Einstein.idade);
   printf("A idade de Newton seria %d  \n", Newton.idade);

   getchar();
   return 0;
}
```

Exemplo em C++ – Versão 3 C

```
#include <stdio.h>

struct Pessoa
{
   int dia;
   int mes;
   int ano;
   int idade;
};
```

```
void Calc_Idade(struct Pessoa& p, int dia, int mes, int ano)
{
   p.idade = ano - p.ano;

   if (p.mes > mes)
   {
        p.idade = p.idade - 1;
   }
   else
   {
        if ( p.mes == mes)
        {
            if (p.dia > dia)
            {
                p.idade = p.idade  - 1;
            }
        }
   }

}
```

```
int main()
{
   struct Pessoa Einstein, Newton;

   Einstein.dia    = 14;
   Einstein.mes  = 3;
   Einstein.ano  = 1879;
   Einstein.idade = -1;

   Newton.dia    = 4;
   Newton.mes  = 1;
   Newton.ano   = 1643;
   Newton.idade = -1;

   Calc_Idade ( Einstein, 28, 8, 2009);
   Calc_Idade ( Newton, 28, 8, 2009);

   printf("A idade de Einstein seria %d \n", Einstein.idade);
   printf("A idade de Newton seria %d  \n", Newton.idade);

   getchar();
   return 0;
}
```

Funciona só em C++!
Referência escondida!

Exemplo em C++ – Simples

```
#include <stdio.h>

struct Pessoa
{
   int diaP;
   int mesP;
   int anoP;
   int idadeP;

   // A função está dentro da estrutura ou struct
   void Calc_Idade ( int diaAT, int mesAT, int anoAT)
   {
      idadeP = anoAT - anoP;
      if (mesP <  mesAT)
      {
         idadeP = idadeP - 1;
      }
      else
      {
          if (mesP == mesAT)
          {
             if (diaP > diaAT)
             {
                idadeP = idadeP - 1;
             }
          }
      }
   }
};
```

```
int main()
{
   struct Pessoa Einstein, Newton;

   Einstein.diaP   = 14;
   Einstein.mesP = 3;
   Einstein.anoP  = 1879;

   Newton.diaP    = 4;
   Newton.mesP  = 1;
   Newton.anoP   = 1643;

   Einstein.Calc_Idade ( 24, 8, 2009 );
   Newton.Calc_Idade ( 24, 8, 2009 );

   printf("A idade de Einstein seria %d \n", Einstein.idadeP);
   printf("A idade de Newton seria %d   \n", Newton.idadeP);

   getchar();
   return 0;
}
```

Exemplo em C++ – Construtora

```
#include <stdio.h>

struct Pessoa
{
public:
   int diaP;
   int mesP;
   int anoP;
   int idadeP;

   // Esta é uma função construtora.
   // Uma construtora inicializa variáveis
   Pessoa (int diaNa, int mesNa, int anoNa)
   {
      diaP    = diaNa;
      mesP  = mesNa;
      anoP   = anoNa;
      idadeP = -1;
   }

   // A função está dentro da estrutura  ou struct
   void Calc_Idade(int diaAT, int mesAT, int anoAT)
   {
      . . .
   }

};
```

```
int main()
{
   Pessoa Einstein ( 14, 3, 1879 );
   Pessoa Newton ( 4, 1, 1643 );

   Einstein.Calc_Idade ( 24, 8, 2009 );
   Newton.Calc_Idade ( 24, 8, 2009 );

   printf("Einstein teria %d \n", Einstein.idadeP);
   printf("Newton teria %d   \n", Newton.idadeP);

   getchar();
   return 0;
}
```

Exemplo em C++ – Construtora

```
#include <stdio.h>

struct Pessoa
{
public:
   int diaP;
   int mesP;
   int anoP;
   int idadeP;

   // Esta é uma função construtora.
   // Uma construtora inicializa variáveis
   Pessoa (int diaNa, int mesNa, int anoNa)
   {
      diaP    = diaNa;
      mesP  = mesNa;
      anoP   = anoNa;
      idadeP = -1;
   }

   // A função está dentro da estrutura  ou struct
   void Calc_Idade(int diaAT, int mesAT, int anoAT)
   {
      . . .
   }

};
```

```
int main()
{
   Pessoa Einstein.Pessoa ( 14, 3, 1879 );
   Pessoa Newton.Pessoa ( 4, 1, 1643 );

   Einstein.Calc_Idade ( 24, 8, 2009 );
   Newton.Calc_Idade ( 24, 8, 2009 );

   printf("Einstein teria %d \n", Einstein.idadeP);
   printf("Newton teria %d   \n", Newton.idadeP);

   getchar();
   return 0;
}
```

Erro

Exemplo em C++ – .h e .cpp

```
#include <stdio.h>

struct Pessoa
{
public:
   int diaP;
   int mesP;
   int anoP;
   int idadeP;

   // Esta é uma função construtora.
   // Uma construtora inicializa variáveis
   Pessoa(int diaNa, int mesNa, int anoNa)
   {
      diaP   = diaNa;
      mesP   = mesNa;
      anoP   = anoNa;
      idadeP = -1;
   }

   // A função está dentro da estrutura  ou struct
   void Calc_Idade(int diaAT, int mesAT, int anoAT)
   {
      . . .
   }
};
```

Pessoa.h

```
#include "Pessoa.h"

int main()
{
   Pessoa Einstein( 14, 3, 1879 );
   Pessoa Newton( 4, 1, 1643 );

   Einstein.Calc_Idade( 24, 8, 2009);
   Newton.Calc_Idade( 24, 8, 2009);

   printf("Einstein teria %d \n", Einstein.idadeP);
   printf("Newton teria %d   \n", Newton.idadeP);

   getchar();
   return 0;
}
```

Main.cpp

Obs.: Um elemento publico (*public*) é acessível em qualquer parte do programa, por exemplo no main() ou em qualquer função

Em uma estrutura (*struct*) do C++, todos os elementos são públicos (*public*) por definição. Mas isto não impede de se explicitar usando a palavra *public:* antecedendo os elementos.

Private X Public

```
#include <stdio.h>

struct Pessoa
{
private:
   int diaP;
   int mesP;
   int anoP;
   int idadeP;

public:
  Pessoa(int diaNa, int mesNa, int anoNa)
   {
      . . .
   }

   void Calc_Idade(int diaAT, int mesAT, int anoAT)
   {
      . . .
   }

   /* muitos programadores chamariam esta função
      abaixo de getIdade()  */

   int informaldade()
   {
       return idadeP;
   }
};
```

Pessoa.h

```
#include "Pessoa.h"

int main()
{
   Pessoa Einstein ( 14, 3, 1879 );
   Pessoa Newton ( 4, 1, 1643 );

   Einstein.Calc_Idade ( 24, 8, 2009 );
   Newton.Calc_Idade ( 24, 8, 2009 );

   printf("Einstein teria %d \n", Einstein.informaldade());
   printf("Newton teria %d   \n", Newton.informaldade());

   getchar();
   return 0;
}
```

Main.cpp

Algo privado (***private***) em uma estrutura C++ (*struct*) é acessível apenas no escopo dela.

Algo público (***public***) em uma estrutura é assessível por qualquer elemento do programa, por exemplo no *main* ou em outra função.

Exemplo em C++ – Class

```
#include <stdio.h>

class Pessoa
{
private:
   int diaP;
   int mesP;
   int anoP;
   int idadeP;
public:
  Pessoa(int diaNa, int mesNa, int anoNa)
   {
      . . .
   }
   void Calc_Idade(int diaAT, int mesAT, int anoAT)
   {
      . . .
   }
   int informaldade()
   {  . . .
   }
};
```

Pessoa.h

No lugar de ***struct***, em C++, é mais usual utilizar uma ***class***.

Uma primeira diferença entre elas é que em uma ***class*** os elementos são ***private*** por definição.

```
#include "Pessoa.h"
int main()
{
   Pessoa Einstein ( 14, 3, 1879 );
    Pessoa Newton ( 4, 1, 1643 );

    Einstein.Calc_Idade ( 24, 8, 2009 );
    Newton.Calc_Idade ( 24, 8, 2009 );

    printf("Einstein teria %d \n", Einstein.informaldade());
    printf("Newton teria %d   \n", Newton.informaldade());

    getchar();
    return 0;
}
```

Main.cpp

Uma ***class*** (classe) representa a estrutura geral de um conjunto de elementos. Por exemplo, a **class Pessoa** diz que qualquer pessoa terá uma idade e uma 'data' de nascimento...

Cada elemento criado a partir de uma **class** se constitui em um ***object*** (objeto). Exemplos de objetos da classe **Pessoa** são os objetos **Einstein** e **Newton**.

Class – Attributes and Methods

Attributes: as "variáveis" de uma classe são chamadas de atributos.

Methods: as "funções" de uma classe são chamadas de métodos, funções-membro ou ainda operações.

```
#include <stdio.h>

class Pessoa
{
private:

   int diaP;
   int mesP;
   int anoP;
   int idadeP;

public:

  Pessoa(int diaNa, int mesNa, int anoNa)
   {
      . . .
   }

   void Calc_Idade(int diaAT, int mesAT, int anoAT)
   {
      . . .
   }

   int informaIdade()
   {
     . . .
   }

};
```

Classe dividida em .h e .cpp

Pessoa.h

```
#include <stdio.h>

class Pessoa
{
private:
   int diaP;
   int mesP;
   int anoP;
   int idadeP;

public:
  Pessoa(int diaNa, int mesNa, int anoNa);
  void Calc_Idade(int diaAT, int mesAT, int anoAT);
  int informaIdade();    // int getIdade();
};
```

Pessoa.cpp

```
#include "Pessoa.h“
Pessoa::Pessoa(int diaNa, int mesNa, int anoNa)
{
   ...
}

void Pessoa::Calc_Idade(int diaAT, int mesAT, int anoAT)
{
  . . .
 }

int Pessoa::informaIdade ( )          // int getIdade()
{
   return idadeP;
}
```

main.cpp

```
#include "Pessoa.h"

int main()
{  Pessoa Einstein(14, 3, 1879);
   Pessoa Newton(4, 1, 1643);

   Einstein.Calc_Idade ( 24, 9, 2009);
   Newton.Calc_Idade ( 24, 9, 2009);

   printf("Einstein teria %d \n", Einstein.informaIdade());
   printf("Newton teria %d   \n", Newton.informaIdade());

   getchar();
   return 0;
}
```

Entendendo o encapsulamento

Exercício: fazer com que a “impressão” (*printf...*) da idade ocorra dentro dos próprios objetos, programando isto no método apropriado da classe Pessoa...

```
#include "Pessoa.h"
int main()
{
    Pessoa Einstein ( 14, 3, 1879 );
    Pessoa Newton ( 4, 1, 1643 );

    Einstein.Calc_Idade ( 24, 8, 2009 );
    Newton.Calc_Idade ( 24, 8, 2009 );

    printf ( "Einstein teria %d \n", Einstein.informaldade() );
    printf ( "Newton teria %d   \n", Newton.informaldade() );

    getchar();
    return 0;
}
```