

Classes e objetos

Roberto Rocha

Programação Orientada a Objetos

Linguagem C e C++

Saída de dados

C

```
#include <stdio.h>
int main() {
    printf("Alo, mundo!");
    return 0;
}
```

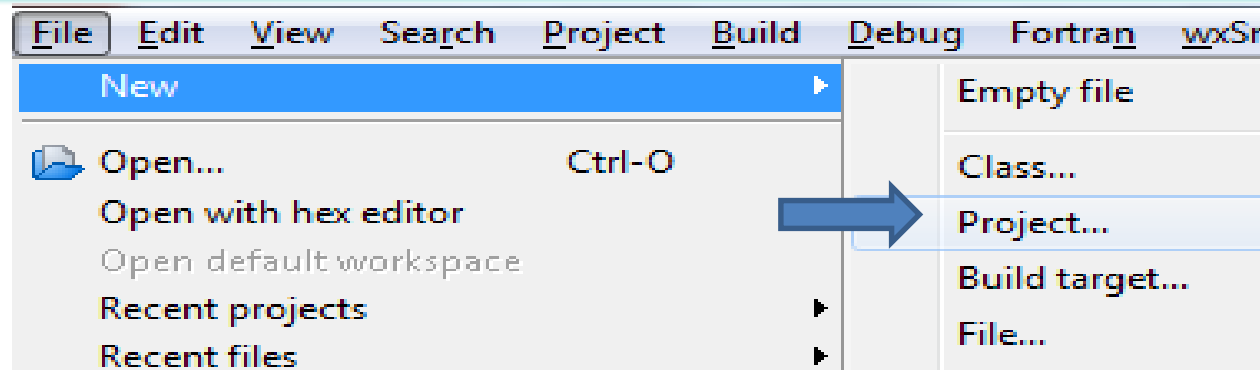
C++

```
#include <iostream>

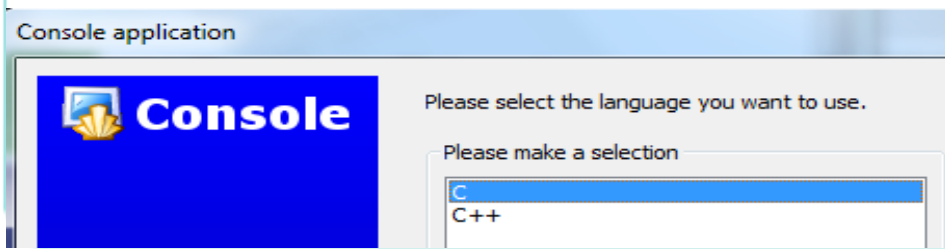
using namespace std;

int main() {
    cout << "Alo, mundo!";
    return 0;
}
```

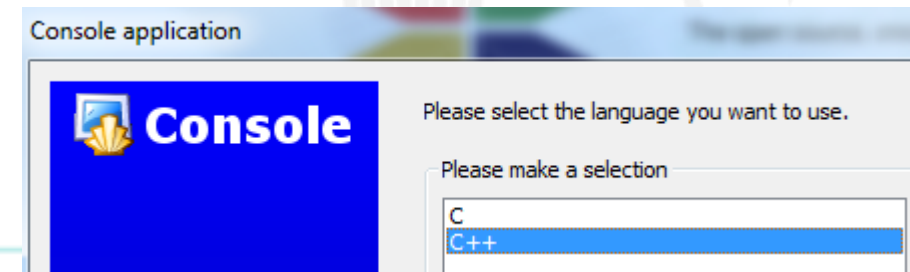
Linguagem C e C++



em seguida escolha o tipo de projeto
e a linguagem : C ou C++



ou



Linguagem C e C++

As linguagens C e C++ são diferentes, apesar de realizarem as mesmas ações e utilizarem o mesmo programa compilador.

Elas possuem peculiaridades, mesmo havendo elementos em comum.

A instrução `#include` faz a chamada de uma biblioteca de recursos externos.

Na linguagem C, o fluxo de saída é controlado pela função `printf()`, que está na biblioteca `stdio.h`; na linguagem C++, o fluxo de saída é controlado pela instrução `cout <<`, que está na biblioteca `iostream`.

O símbolo `<<`, chamado operador de inserção, indica que a mensagem está sendo direcionada para o fluxo de saída `cout`.

Em C++, existe a instrução **using namespace std**, que indica a área de memória (namespace) em que o programa deve ser usado (`using`);

Linguagem C e C++

Entrada de dados

.C

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a,b,soma;
    printf("Digite o valor de a:");
    scanf("%i",&a);
    printf("Digite o valor de b:");
    scanf("%i",&b);
    soma=a+b;
    printf("A soma de %i + %i = %i\n",a,b,soma);
    return 0;
}
```

```
Digite o valor de a:2
Digite o valor de b:3
A soma de 2 + 3 = 5
```

.CPP

```
#include <iostream>
#include <string.h>
using namespace std;

int main()
{
    int a,b,soma;
    cout << "Digite o valor de a:";
    cin >> a;
    cout << "Digite o valor de b:";
    cin >> &b;
    soma=a+b;
    cout << "A soma de "<< a << " + " << b << " = " << soma << endl;
    return 0;
}
```

Algoritmos POO

Os trabalhos que realizamos até aqui tem o inconveniente de que os tipos de dados e os procedimentos ficam separados e isso exige que cada procedimento manipule corretamente os dados. Qualquer alteração na representação dos dados exige manutenção em todos os procedimentos que manipulem esses dados.

O paradigma da programação orientada a objetos (POO) implica em qualquer entidade ter representados características e comportamentos.

Vantagens:

- reutilização de código
- confiabilidade
- facilidade de manutenção e extensão.

Algoritmos POO

Classe

No mundo real, podemos agrupar, por exemplo, animais, automóveis. Quando realizamos esses agrupamentos, por exemplo Classes de animais, conjuntos de tipos de animais que possuem comportamentos e características semelhantes.

Exemplos:

Classe mamífero

características:

corpo coberto por pelos - total ou parcialmente

peso

aquáticos

comportamentos:

produzem leite

são endodérmicos

locomovem



Algoritmos POO

Classe

Classe Carro

características:

Cor

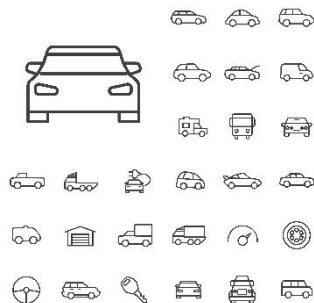
Marca

Modelo

comportamentos:

Gastam combustível

Percorrem distâncias



Classe Funcionário

características:

Atributos

Sexo

Salário

Nome

Endereço

Data de Nascimento

Cargo

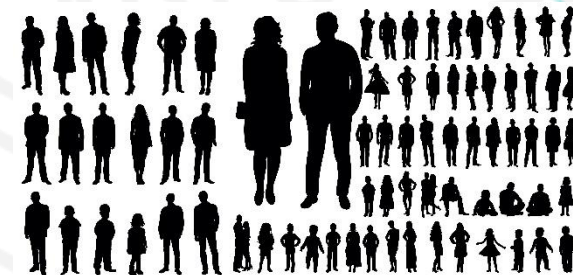
comportamentos:

Métodos

Recebem salários

Recebem férias

Recebem décimo terceiro



Uma classe é um tipo abstrato de dados que possui características e operações, na POO denominados atributos e métodos, respectivamente.

POO – Classe em C++

Classe

Uma classe em C++ é uma extensão do registro (**struct**) em C, adicionando-se a possibilidade de definir componentes que são funções (**métodos**).

Sintaxe de uma classe:

```
class nome_classe
{
    especificador_de_acesso:
    declaração_dos_atributos;
    declaração_dos_métodos;
}
```

Onde:

Nome_classe: é o identificador da classe.

Especificador_de_acesso pode ser **public**, **private** ou **protect**:

public: permite que atributos e métodos sejam acessados por qualquer classe.

private: permite que atributos e métodos sejam acessados (visíveis) somente dentro da classe em que foram declarados, o que garante o encapsulamento, uma das vantagens de POO.

Por definição padrão, os atributos e métodos são **private**.

protect: funciona como o **private**, mas permite o acesso também pelas classes derivadas, ou seja, pelas denominadas subclasses. (subclasse e superclasse serão mostrados a diante)

Dentro de uma classe é possível ter atributos e métodos definidos com especificadores de acesso diferentes.

declaração_dos_atributos: local onde os atributos são definidos com especificador de acesso, tipo e nome.

declaração_dos_métodos: local onde os métodos são definidos com especificador de acesso, tipo de retorno, nome e lista de parâmetros.

POO – Classe em C++

Exemplo:

```
class Produto
{
    // definição dos atributos e tipo que por definição são private
    float valor;
    char tipo;
    // definição do método valorImposto que por definição é private
    float valorImposto(float pre)
};
```

POO – Classe em C++

Exemplo:

```
class Pessoa
{
    // definição dos atributos nome, sexo e ano de nascimento definidos no escopo public
    public:
    char nome[30];
    char sexo;
    int ano;
    // definição do atributo salario definido no escopo private
    private:
    float salario;
    //definição do método calcularAumento definido no escopo private
    private:
    void calcularAumento();

    public:
    // definição dos métodos calcularIdade e calcularFilhos definidos no escopo public
    int calcularIdade(int ano);
    void calcularFilhos(char sexo);
};
```

POO – Objeto

Um objeto é um indivíduo único dentro do conjunto que é a classe.

Exemplos:

classe mamífero

características:

corpo coberto por pelos - total ou parcialmente

peso

aquáticos

comportamentos:

produzem leite

são endodérmicos

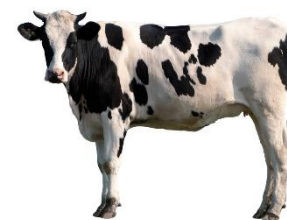
locomovem

Objetos:

vacaMimosa

leoaBranca

baleiaSapeca



classe Carro

características:

Cor

Marca

Modelo

comportamentos:

Gastam combustível

Percorrem distâncias

Objetos:

golDoChico

fiatUnoDoTio



POO – Objeto

Um objeto é um indivíduo único dentro do conjunto que é a classe.

Classe Funcionário

características:

- Sexo
- Salário
- Nome
- Endereço
- Data de Nascimento
- Cargo

comportamentos:

- Recebem salários
- Recebem férias
- Recebem décimo terceiro



Objetos:

joaoSilva
mariaJose

Quando um programa faz uso de uma variável, ela precisa ser declarada para que espaços de memória sejam alocados e ela seja utilizada no processamento. Com os objetos, existe essa mesma exigência. Os objetos precisam ser declarados e instanciados para que existam e possam ser utilizados.

INSTANCIANDO OBJETOS EM C++

SINTAXE:

```
nome_da_classe nome_do_objeto;
```

Exemplo:

```
Produto p;
```

Um objeto denominado p, que representa um elemento da classe Produto, foi declarado e instanciado. Para acessar os atributos públicos do objeto, é necessário seguir a sintaxe:

```
nome_do_objeto.atributo
```

Se desejarmos executar um método que tenha o especificador de acesso publico, é necessário seguir a sintaxe:

```
nome_do_objeto.nome_do_método(parâmetro)
```

Vamos ver um exemplo completo:

INSTANCIANDO OBJETOS EM C++

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
class Produto
{
public:
    // atributos nome,valor
    char nome[20];
    float valor;
    //método que calcula o valor do imposto
    float valorImposto()
    {
        float vi;
        if (valor<500)
            vi= valor*10/100;
        else
            vi=valor*15/100;
        return vi;
    }
};
```

```
int main()
{
    Produto p;
    float preco,imp;
    char nomeProduto[30];
    printf("digite o nome do produto:");
    gets(nomeProduto);
    printf("digite o preço do produto:");
    scanf("%f",&preco);
    strcpy(p.nome,nomeProduto);
    p.valor=preco;
    printf("Dados do produto\n");
    printf("\nnome:%s",p.nome);
    printf("\nPreço:%5.2f",p.valor);
    printf("\nValor Imposto:%5.2f\n",p.valorImposto());
    system("pause");
    return 0;
}
```

```
digite o nome do produto:caneta
digite o preço do produto:10
Dados do produto

nome:caneta
Preço:10.00
Valor Imposto: 1.00
Pressione qualquer tecla para continuar. . .
```

```
digite o nome do produto:caneta
digite o preço do produto:1000
Dados do produto

nome:caneta
Preço:1000.00
Valor Imposto:150.00
Pressione qualquer tecla para continuar. . .
```


INSTANCIANDO OBJETOS EM C++

Implementação das funções externas a declaração da classe

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
class Produto
{
public:
    // atributos nome,valor
    char nome[20];
    float valor;
    //método que calcula o valor do imposto
    float valorImposto();
};
float Produto::valorImposto()
{
    float vi;
    if (valor<500)
        vi= valor*10/100;
    else
        vi=valor*15/100;
    return vi;
}
```

```
int main()
{
    Produto p;
    float preco,imp;
    char nomeProduto[30];
    printf("digite o nome do produto:");
    gets(nomeProduto);
    printf("digite o preço do produto:");
    scanf("%f",&preco);
    strcpy(p.nome,nomeProduto);
    p.valor=preco;
    printf("Dados do produto\n");
    printf("\nnome:%s",p.nome);
    printf("\nPreço:%5.2f",p.valor);
    printf("\nValor Imposto:%5.2f\n",p.valorImposto());
    system("pause");
    return 0;
}
```

```
digite o nome do produto:caneta
digite o preço do produto:10
Dados do produto
nome:caneta
Preço:10.00
Valor Imposto: 1.00
Pressione qualquer tecla para continuar. . .
```

```
digite o nome do produto:caneta
digite o preço do produto:1000
Dados do produto
nome:caneta
Preço:1000.00
Valor Imposto:150.00
Pressione qualquer tecla para continuar. . .
```

INSTANCIANDO OBJETOS EM C++

Exercício:

Crie uma classe e instancie um objeto para a classe carro.

Atributos: marca, cor, anoFabricacao, valor

Método IPVA:

5% do valor do carro - 10% por ano de uso.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  class Produto
5  {
6  public:
7      // atributos nome, valor
8      char nome[20];
9      float valor;
10     // método que calcula o valor do imposto
11     float valorImposto()
12     {
13         float vi;
14         if (valor < 500)
15             vi = valor * 10 / 100;
16         else
17             vi = valor * 15 / 100;
18         return vi;
19     }
20 };
21 int main()
22 {
23     Produto p;
24     float preco, imp;
25     char nomeProduto[30];
26     printf("digite o nome do produto:");
27     gets(nomeProduto);
28     printf("digite o preço do produto:");
29     scanf("%f", &preco);
30     strcpy(p.nome, nomeProduto);
31     p.valor = preco;
32     printf("Dados do produto\n");
33     printf("\nnome: %s", p.nome);
34     printf("\nPreço: %5.2f", p.valor);
35     printf("\nValor Imposto: %5.2f\n", p.valorImposto());
36     system("pause");
37     return 0;
38 }
```

INSTANCIANDO OBJETOS EM C++

Crie uma classe e instancie um objeto para a classe carro.

Atributos: marca, cor, anoFabricacao, valor

Método IPVA:

5% do valor do carro - 10% por ano de uso.

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;

class Carro {
public:
    char marca[20];
    string cor;
    int ano;
    float valor;
    float valorIpva (int anoRef){
        float ipva;
        ipva = valor*(float) 5/100;
        ipva = ipva - ((10.0*(anoRef-ano)*ipva));
        if (ipva<0){
            ipva=0;
        }
        return ipva;
    }
};
```

```
int main()
{
    Carro c1;
    printf("Digite a Marca do carro:");
    gets(c1.marca);
    printf("Digite a Cor do carro:");
    cin >> c1.cor;
    printf("Digite a Ano de compra carro:");
    cin >> c1.ano;
    printf("Digite o preco de compra carro:");
    cin >> c1.valor;
    cout <<"Dados do carro:"<< endl;
    cout << "marca:"<< c1.marca<<endl;
    cout << "cor:"<< c1.marca<<endl;
    cout << "ano:"<< c1.ano<<endl;
    cout << "valor:"<< c1.valor<<endl;
    cout << "IPVA:"<< c1.valorIpva(2020) <<endl;
    system("pause");
    return 0;
}
```

Construtores e destrutores

O método construtor é executado uma única vez para cada objeto e é utilizado para reservar os espaços de memória de acordo com o tipo de cada atributo.

Esse método pode receber parâmetros, mas **não pode retornar valores**.

Quando esse método não é explicitamente definido, é executado o construtor padrão, que apenas aloca espaços em memória.

Quando o método construtor é declarado, ele deve ter o mesmo nome da classe e se torna necessário definir também o método destrutor, que é utilizado para liberar os espaços de memória utilizados pelos atributos.

O método destrutor **não pode possuir parâmetros e nem retorno** e deve ter o mesmo nome da classe precedido pelo **símbolo ~**

Vamos ver um exemplo completo:

Construtores e destrutores

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
class Produto
{ public:
```

```
    char nome[20];
```

```
    float valor;
```

```
    //metodo construtor
```

```
    Produto ()
```

```
    { printf("\nexecutando o método construtor\n");
```

```
      printf("digite o nome do produto:");
```

```
      gets(nome);
```

```
      printf("digite o preço do produto:");
```

```
      scanf("%f",&valor);
```

```
    }
```

```
    // metodo destrutor
```

```
    ~Produto()
```

```
    { printf("\n\nExecutando o método destrutor\n");
```

```
      printf("o produto %s foi destruido!\n",nome);
```

```
    }
```

```
    //método que calcula o valor do imposto
```

```
    float valorImposto()
```

```
    { float vi;
```

```
      if (valor<500)
```

```
        vi= valor*10/100;
```

```
      else
```

```
        vi=valor*15/100;
```

```
      return vi;
```

```
    }
```

```
    //método que imprime o produto
```

```
    void imprimeProduto()
```

```
    { printf("Dados do produto\n");
```

```
      printf("\nnome:%s",nome);
```

```
      printf("\nPreço:%5.2f",valor);
```

```
      printf("    ,valorImposto());
```

```
    }
```

```
};
```

```
int main()
{ Produto p;
  p.imprimeProduto();
  system("pause");
  return 0;
}
```

```
executando o método construtor
digite o nome do produto:Caneta
digite o preço do produto:10.00
Dados do produto

nome:Caneta
Preço:10.00
Valor Imposto: 1.00
Pressione qualquer tecla para continuar. . .

Executando o método destrutor
o produto Caneta foi destruido!

Process returned 0 (0x0)   execution time : 22.681 s
Press any key to continue.
```

Construtores e destrutores

Exercício:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
class Produto
{ public:
    char nome[20];
    float valor;
    //metodo construtor
    Produto ()
    { printf("\nexecutando o método construtor\n");
      printf("digite o nome do produto:");
      gets(nome);
      printf("digite o preço do produto:");
      scanf("%f",&valor);
    }
    // metodo destrutor
    ~Produto()
    { printf("\n\nExecutando o método destrutor\n");
      printf("o produto %s foi destruido!\n",nome);
    }
    //método que calcula o valor do imposto
    float valorImposto()
    { float vi;
      if (valor<500)
        vi= valor*10/100;
      else
        vi=valor*15/100;
      return vi;
    }
    //método que imprime o produto
    void imprimeProduto()
    { printf("Dados do produto\n");
      printf("\nnome:%s",nome);
      printf("\nPreço:%5.2f",valor);
      printf("    ,valorImposto());
    }
};
```

```
int main()
{ Produto p;
  p.imprimeProduto();
  system("pause");
  return 0;
}
```

Crie os métodos construtor e destrutor para a classe carro, e um programa principal para instanciar um objeto da classe.

Atributos: marca, cor, anoFabricacao, valor
Método IPVA:
5% do valor do carro - 10% por ano de uso.

Construtores e destrutores

Pode-se também ter vários tipos de construtores!!

Por exemplo vamos criar um construtor que receba como parâmetros os valores de nome e valor.

Caso se deseje pode utilizar o construtor que peça os valores ou o construtor que inclua valores nos atributos nome e valor.

Veja:

Construtores e destrutores

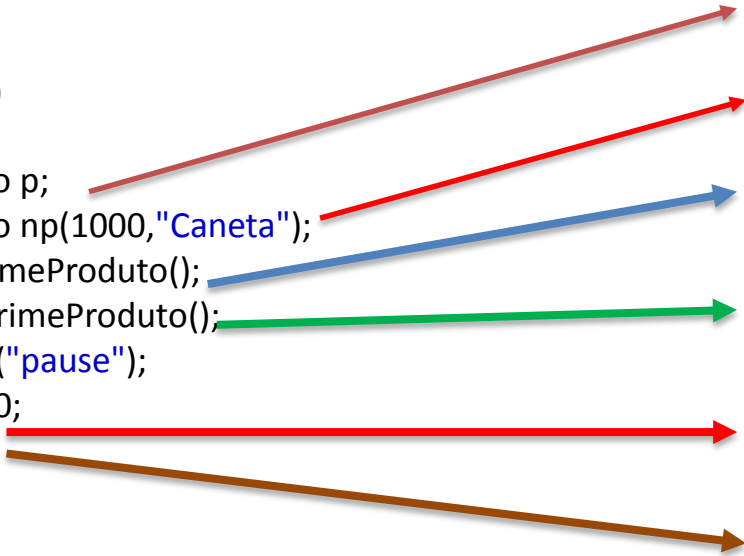
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
class Produto
{
public:
    char nome[20];
    float valor;
    //metodo construtor sem parametros
    Produto ();
    //metodo construtor com parametros
    Produto (float v, const char n[20]);
    // metodo destrutor
    ~Produto();
    //método que calcula o valor do imposto
    float valorImposto();
    //método que imprime o produto
    void imprimeProduto();
};
```

```
//metodo construtor sem parametros
Produto::Produto()
{
    printf("\nexcutando o método construtor sem parametros\n");
    printf("digite o nome do produto:");
    gets(nome);
    printf("digite o preço do produto:");
    scanf("%f",&valor);
}
//metodo construtor com parametros
Produto::Produto (float v, const char n[20])
{
    printf("\nexcutando o método construtor com parametros\n");
    strcpy(nome,n);
    valor=v;
}
// metodo destrutor
Produto::~~Produto()
{
    printf("\n\nExecutando o método destrutor\n");
    printf("o produto %s foi destruido!\n",nome);
}
```

```
//método que calcula o valor do imposto
float Produto::valorImposto()
{
    float vi;
    if (valor<500)
        vi= valor*10/100;
    else
        vi=valor*15/100;
    return vi;
}
//método que imprime o produto
void Produto::imprimeProduto()
{
    printf("Dados do produto\n");
    printf("\nnome:%s",nome);
    printf("\nPreço:%5.2f",valor);
    printf("\nValor Imposto:%5.2f\n",valorImposto());
}
int main()
{
    Produto p,np(1000,"Caneta");
    p.imprimeProduto();
    np.imprimeProduto();
    system("pause");
    return 0;
}
```


Construtores e destrutores

```
int main()
{
    Produto p;
    Produto np(1000,"Caneta");
    p.imprimeProduto();
    np.imprimeProduto();
    system("pause");
    return 0;
}
```



```
executando o método construtor sem parametros
digite o nome do produto:Borracha
digite o preço do produto:3.50

executando o método construtor com parametros
Dados do produto

nome:Borracha
Preço: 3.50
Valor Imposto: 0.35
Dados do produto

nome:Caneta
Preço:1000.00
Valor Imposto:150.00
Pressione qualquer tecla para continuar. . .

Executando o método destrutor
o produto Caneta foi destruido!

Executando o método destrutor
o produto Borracha foi destruido!

Process returned 0 (0x0)   execution time : 377.668 s
Press any key to continue.
```

Construtores e destrutores

Exercício:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
class Produto
{
public:
    char nome[20];
    float valor;
    //metodo construtor sem parametros
    Produto ();
    //metodo construtor com parametros
    Produto (float v, const char n[20]);
    // metodo destrutor
    ~Produto();
    //método que calcula o valor do imposto
    float valorImposto();
    //método que imprime o produto
    void imprimeProduto();
};
```

```
int main()
{   Produto p;
    p.imprimeProduto();
    system("pause");
    return 0;
}
```

Crie outro métodos construtor agora passando por parâmetros a marca, cor e ano de fabricação e o valor de aquisição do carro, e um programa principal para instanciar um objeto da classe.

Atributos: marca, cor, anoFabricacao, valor

Método IPVA:

5% do valor do carro - 10% por ano de uso.

Trabalhando com vetores de classes

Pode-se trabalhar com um vetor contendo elementos de uma classe.
Exemplo: Criar um vetor para armazenar 10 elementos do tipo Produto

```
int main()
{
    Produto p[10];
    int i;
    // imprimindo os conteudos do vetor
    for (i=0;i<10;i++){
        p[i].imprimeProduto();
    }
    system("pause");
    return 0;
}
```

Crie e imprima 10 objetos da classe carro.

Trabalhando com vetores de ponteiros para classes

Pode-se trabalhar com um vetor contendo elementos de uma classe.
Exemplo: Criar um vetor para armazenar 10 elementos do tipo Produto

```
int main()
{
    Produto p[10];
    int i;
    // imprimindo os conteudos do vetor
    for (i=0;i<10;i++){
        p[i].imprimeProduto();
    }
    system("pause");
    return 0;
}
```

Crie e imprima 10 objetos da classe carro.

Construtores e destrutores

Exercício:

Fazer um algoritmo para:

- Definir uma classe para tratar hora, minutos e segundos;
- Crie um método construtor passando os atributos por parâmetros, e outro sem parâmetros;
- Um método para calcular a diferença entre a dada entre o objeto armazenado e um passado por parâmetro.
- Um método para imprimir um objeto

Crie um programa para instanciar três objetos e imprimir o terceiro objeto como a diferença horária entre os dois primeiros.



PUC Minas
Virtual