

Linguagem de Programação C++

Podemos considerar a implementação desta classe String como um ótimo exercício de revisão.

Devido à relevância da existência de uma classe String com tais características, a linguagem C++ disponibiliza esta classe em sua biblioteca-padrão.

A classe String é definida no arquivo cabeçalho `<string>` e pertence ao namespace `std`.

Ao invés de listar as mensagens que compõem a interface da classe String, vamos analisar um exemplo que explora as principais características desta.

```

// Programa que se utiliza da classe string da biblioteca padrão
#include <iostream>
using std::cout;
using std::endl;
#include <string>
using std::string;
int main()
{
    string s1( "feliz" );
    string s2( " aniversário" );
    string s3;
    // testa operadores de igualdade e relacionais sobrecarregados
    cout << "s1 eh \"\" << s1 << "\"; s2 eh \"\" << s2
        << "\"; s3 eh \"\" << s3 << "\"
        << endl << "O resultado da comparação entre s2 e s1:"
        << endl << "s2 == s1 produz " << ( s2 == s1 ? "verdadeiro" : "falso" )
        << endl << "s2 != s1 produz " << ( s2 != s1 ? "verdadeiro" : "falso" )
        << endl << "s2 > s1 produz " << ( s2 > s1 ? "verdadeiro" : "falso" )
        << endl << "s2 < s1 produz " << ( s2 < s1 ? "verdadeiro" : "falso" )
        << endl << "s2 >= s1 produz " << ( s2 >= s1 ? "verdadeiro" : "falso" )
        << endl << "s2 <= s1 produz " << ( s2 <= s1 ? "verdadeiro" : "falso" );

```

```

// testa a função membro string vazia
cout << "\n\nTestanto s3.empty():" << endl;
if ( s3.empty() )
{
    cout << "s3 eh vazio; atribuindo s1 para s3;" << endl;
    s3 = s1; // atribui s1 a s3
    cout << "s3 agora eh \"\" << s3 << "\"\"";
}
// testa operador de concatenação de string sobrecarregado
cout << "\n\ns1 += s2 produz s1 = ";
s1 += s2; // testa a concatenação sobrecarregada
cout << s1;
// testa operador de concatenação de string sobrecarregado com string
no estilo C
cout << endl << endl << "s1 += \" para voce\" produz" << endl;
s1 += " para voce";
cout << "s1 = " << s1 << endl << endl;
// testa função membro string substr
cout << "A substring de s1 inicializando na localizacao 0 para"
<< endl << "14 caracteres, s1.substr(0, 14), eh:" << endl
<< s1.substr( 0, 14 ) << endl << endl;

```

```

// testa a opção de substr "to-end-of-string"
cout << "A substring de s1 inicializando na" << endl
    << "localizacao 15, s1.substr(15), eh:\n"
    << s1.substr( 15 ) << endl;
// testa o construtor de cópia
string *s4Ptr = new string( s1 );
cout << endl << endl << "*s4Ptr = " << *s4Ptr << endl << endl;
// testa o operador de atribuição (=) com a auto-atribuição
cout << "Atribuindo *s4Ptr para *s4Ptr" << endl;
*s4Ptr = *s4Ptr;
cout << "*s4Ptr = " << *s4Ptr << endl;
// testa o destrutor
delete s4Ptr;
// testa o uso do operador de indexacao para criar lvalue
s1[ 0 ] = 'F';
s1[ 6 ] = 'A';
cout << "\ns1 Depois de s1[0] = 'F' e s1[6] = 'A' eh: "
    << s1 << endl << endl;
//testa o subscrito fora do intervalo com a função membro de string "at"
cout << "Tentativa de atribuir 'd' para s1.at( 30 ) produz:" << endl;
s1.at( 30 ) = 'd'; // ERRO: indice fora do intervalo
return 0; }

```

Linguagem de Programação C++

Polimorfismo

Estudamos detalhadamente um conceito relacionado ao polimorfismo, mais especificamente, o conceito de sobrecarga.

Aprofundaremos, agora, nosso estudo sobre polimorfismo, tratando-o de uma forma mais ampla.

Para tanto, nos utilizaremos do mecanismo de herança, a título de revisão, o mecanismo da herança nos possibilita especializar uma classe genérica (básica).

O polimorfismo nos permite tratar objetos das classes derivadas de forma genérica. Em outras palavras, o polimorfismo nos permite “programar no geral” em vez de “programar no específico”.

Linguagem de Programação C++

Polimorfismo

Inicialmente vamos tratar da possibilidade de sobreposição de uma função membro.

Conforme já vimos, sobreposição é a redefinição de um método definido em uma superclasse em uma de suas subclasses.

O exemplo a seguir demonstra este mecanismo.

```
#include <iostream>
using namespace std;
class Base {
public:
    void func(){
        cout << "Esta eh func() de base" << endl;
    }
};
```

```
class Derivada1: public Base{  
    public:  
        void func(){  
            cout << "Esta eh func() de derivada1" << endl;  
        }  
};  
class Derivada2: public Base{  
    public:  
        void func(){  
            cout << "Esta eh func() de derivada2" << endl;  
        }  
};  
int main(){  
    Base b;  
    Derivada1 d1;  
    Derivada2 d2;  
    b.func();  
    d1.func();  
    d2.func();  
    return 0;  
}
```

Linguagem de Programação C++

Exercício:

O serviço de correio expresso como o SEDEX®, oferece várias opções de entrega, cada qual com custos específicos. Crie uma hierarquia de herança para representar vários tipos de pacote. Utilize Pacote como a classe base da hierarquia, então inclua as classes PacoteDoisDias e PacoteNoite que derivam de Pacote. A classe Pacote deve incluir membros de dados que representam nome, endereço, cidade e CEP tanto do remetente quanto do destinatário do pacote, além dos membros de dados que armazenam peso (em quilos) e custo por quilo para entrega do pacote. O construtor pacote deve inicializar estes membros de dados. Assegure que o peso e o custo por quilo contenham valores positivos. Pacote deve fornecer uma função-membro pública calculaCusto que retorna um double indicando o custo associado com a entrega do pacote. (continua)

A função `calculaCusto` de `Pacote` deve determinar o custo multiplicando o peso pelo custo (em quilos). A classe derivada `PacoteDoisDias` deve herdar a funcionalidade da classe básica `Pacote`, mas também incluir um membro de dados que representa uma taxa fixa que a empresa de entrega cobra pelo serviço de entrega em dois dias. O construtor de `PacoteDoisDias` deve receber um valor para inicializar este membro de dados. `PacoteDoisDias` deve redefinir a função-membro `calculaCusto` para que ela calcule o custo de entrega adicionando a taxa fixa ao custo baseado em peso calculado pela função `calculaCusto` da classe básica `Pacote`. A classe `PacoteNoite` deve herdar diretamente da classe `Pacote` e deve conter um membro de dados adicional para representar uma taxa adicional por quilo cobrada pelo serviço de entrega noturno. `PacoteNoite` deve redefinir a função-membro `calculaCusto` para que ela acrescente a taxa adicional por quilo ao custo padrão por quilo antes de calcular o custo de entrega. Escreva um programa de teste que cria objetos de todos os tipos definidos e testa a função `calculaCusto`.

Exercício:

O serviço de correio expresso como o SEDEX®, oferece várias opções de entrega, cada qual com custos específicos. Crie uma hierarquia de herança para representar vários tipos de pacote. Utilize Pacote como a classe base da hierarquia, então inclua as classes PacoteDoisDias e PacoteNoite que derivam de Pacote. A classe Pacote deve incluir membros de dados que representam nome, endereço, cidade e CEP tanto do remetente quanto do destinatário do pacote, além dos membros de dados que armazenam peso (em quilos) e custo por quilo para entrega do pacote. O construtor pacote deve inicializar estes membros de dados. Assegure que o peso e o custo por quilo contêm valores positivos. Pacote deve fornecer uma função-membro pública calculaCusto que retorna um double indicando o custo associado com a entrega do pacote. A função calculaCusto de Pacote deve determinar o custo multiplicando o peso pelo custo (em quilos). A classe derivada PacoteDoisDias deve herdar a funcionalidade da classe básica Pacote, mas também incluir um membro de dados que representa uma taxa fixa que a empresa de entrega cobra pelo serviço de entrega em dois dias. O construtor de PacoteDoisDias deve receber um valor para inicializar este membro de dados. PacoteDoisDias deve redefinir a função-membro calculaCusto para que ela calcule o custo de entrega adicionando a taxa fixa ao custo baseado em peso calculado pela função calculaCusto da classe básica Pacote. A classe PacoteNoite deve herdar diretamente da classe Pacote e deve conter um membro de dados adicional para representar uma taxa adicional por quilo cobrada pelo serviço de entrega noturno. PacoteNoite deve redefinir a função-membro calculaCusto para que ela acrescente a taxa adicional por quilo ao custo padrão por quilo antes de calcular o custo de entrega. Escreva um programa de teste que cria objetos de todos os tipos definidos e testa a função calculaCusto.