Lucas Santos – GUIA C++

CONSIDERAÇÕES IMPORTANTES

#include iostream -> Importa as bibliotecas basicas do C++.

#include string -> Importa a biblioteca string e permite utilizar variaveis string.

#include algorithm -> Importa a biblioteca algorithm que possue funções com reverse(), q inverte uma string.

#include fstream -> Importa a biblioteca que permite abrir um arquivo.

#include sstream -> Importa a biblioteca que permite utilizar o stringstream.

using namespace std; -> Permite utilizar os atributos de C++ sem explicitar o
std - Exemplo std cout

C++ é uma linguagem case sensitive -> Se utilziar letras maiusculas no lugar de minusculas ou o contrário, resultará em erro.

Utilizar para separar os elementos -> Exemplo cout Hoje e dia variavelDia;

Um ponteiro (ex int nome_ponteiro**) NÃO PODE ter o mesmo nome de uma variavel.**
- int i = 10; int *i; - NAO PODE EXISTIR
- int i = 10; int *ii = &i - PODE EXISTIR

Tipos de variavel -> int i - armazena um valor inteiro.
&i - representa a posição de i na memória;
int ponteiro = &i - armazena o valor presente na posição &i na memoria.

Um poteiro armazena o primeiro valor do array -> int array[] = {10, 20}; int ponteiro = array;
ponteiro vai ser igual a 10;
cout ponteiro - retorna 10
cout (ponteiro + 1) - retorna 20

É possivel utilizar o valor de uma variavel para alocar memoria para um array
Exemplo: int tamanho = 10;
int array = new int[tamanho] - o array possui 10 posições

size_t -> tipo de variavel não negativa que por padrão representa tamanhos e indices
Ele é o tipo padrão retornado por funções como string.length() ou string.find().

String::npos -> É um valor especial utilizado para demonstrar que nada foi encontrado.
Quando você faz nome.find(x) e não encontra x, o retorno é string::npos.

COMANDOS DA LINGUAMGEM

endl -> pula linha

cout -> imprime coisas na tela. Exemplo cout ola mundo;

cin -> recebe valores a partir de entradas do usuário.
Exemplo cout Digite o dia ;
cin variavel;

Declaração de arrays -> int array[] = {} ou int array;
int array = new int[5]; -> O array possui 5 posições.

#define -> #define é utilizado para definir um macro no sistema, tudo que for definido com isso valera para o programa inteiro
Exemplo: #define LIN = 10; -> Define que em todas as ocorrências de LIN sera LIN = 10.
int array = new int[LIN] -> (o array tera 10 posições).

Sleep(numero) -> faz o tempo da execução parar pelo tempo indicado.

Classes - POO

Classes são utilizadas como um molde para criar objetos no programa. Ela agrupa variáves(chamadas de atributos) e funções(chamadas de metodos) que fazem sentido juntas. Os atributos são como características do objeto, e os métodos, ações que o memso pode tomar. Como por exemplo: uma classe denominada como Carro, cujos atributos podem ser declarados como a sua cor e modelo. Enquanto seus métodos podem ser declarados como acelerar() ou frear().

Criação da classe -> Para criar uma classe – definir que é uma classe com a palavra reservada **class** e o nome da classe. Exemplo:

```
class Carro{
        //código...

}
```

private e public -> A classe é separada em private e public.

private: -> Elementos que so podem ser acessados na classe(aumenta a segurança da aplicação – geralmente atributos são declarados como private).

public: -> Elementos que pode ser acessados de fora da classe.

Exemplo:

```
class Carro{
        private:
        string cor;  //atributo privado
        string modelo;  //atributo privado

        public:
        // aqui vem os métodos da classe (funções)

};
```

getter e setter -> Funcionam para "pegar" (get) e "setar"(set) valores a atributos.

Servem para controlar como o dado é acessado ou alterado nos atributos, proteger os dados do programa e manter o programa organizado .

Exemplo:

```
class Carro{
        private:
        string cor;
        string modelo;

        public:
        void setCor(string cor){
        this-> cor = cor;
        }
        Void setModelo(string modelo){
        This->modelo = modelo;
        }

        string getCor(){
        return cor;
        }
        string getModelo(){
        return modelo;
        }

};
```

Método Construtor -> É uma função especial que é chamada automaticamente quando um objeto da classe é criado. É recomendado criar sempre 2 construtores. O construtor deve **possuir o mesmo nome da classe.**

1) **Sem parâmetro(vazio) ->** Serve para definir um valor padrão para os atributos caso nada for passado.
2) **Com parâmetros ->** Serve para atribuir valor aos atributos da da classe.
3) **this ->** É utilizado para refenciar um atributo na classe. Comumente utilizado quando o parâmetro passado possui o mesmo nome do atributo ou em outros casos para refenciar diretamente o atributo.

Exemplo:

```
class Carro{
        private:
        string cor;
        string modelo;

        public:
        //construtor sem parâmetros(vazio)
        public Carro(){
        cor = “padrao”;
        modelo = “basico”;
        }
        //construtor com parâmetros
        public Carro(string cor, string modelo){
        this->cor = cor;  //this->cor  utilizado para referenciar qual é o atributo pois
                          //existe o atributo cor e o parâmetro string cor com o mesmo nome.
        This->modelo = modelo;
        }
};
```

Método funcional -> Dentro de classes, é possível construir métodos funcionais que são responsáveis por servirem como as possíveis ações que uma classe pode realizar.

Exemplo:

```
Class Carro{
        public:
        void acelerarCarro(){  //exemplo sem passar parâmetros
        cout << "O carro está acelerando" << endl;
        }
        string exibeCarro(string cor, string modelo){  //exemplo passando parâmetros
        string especificacao = cout << ''A cor e: " + cor + " e o modelo e: " + modelo << endl;
        return especificacao;
        }
};
```

Utilizando a classe -> Para utilizar uma classe, seus atributos e métodos, é necessário que ela seja "chamada" no método principal main. Utilzando da criação de um objeto na classe principal main.

Exemplo:

```
int main(int argc, char** argv)
{
        Carro *carro; // objeto carro criado sem a passagem de parâmetros( construtor vazio)
                        // é possível utilizar seus métodos, mas no momentos seus atributos
                        // não possuem valor atribuído.
        carro.setCor("Azul"); // agora uma cor foi atribuída
        carro.setModelo("Civic"); // agora um modelo foi atribuido

        Carro *carro2 = new Carro("Rosa", "BMW");  // agora os parâmetros foram passados
                                                    // no construtor;
        carro2->exibeCarro();

        return 0;
}
```

FUNÇÕES DO C++

.length()-> retorna o tamanho da string (.size faz a mesma coisa)

.**erase() ->** remove um caractere.

Exemplo string palavra = teste;

palavra.erase(1, 1); - Remove 1 caractere na posição 1 ('e')

reverse(variavel.begin(), variavel.end) -> inverte uma string.

ifstream arquivo(arquivo.txt); -> abre um arquivo txt ja existente para lê-lo no programa.

.find_last_of(condição) -> encontra a ultima ocorrencia da condição na string.

.**substr(começo, fim) ->** captura um trecho da string do inicio marcado ate um segundo ponto.

Ex string nome = Lucas Santos;

string primeiroNome = nome.substr(0, nome.find(' '));

Vai retornar o Nome até o primeiro espaço.

stringstream nome_qualquer(string) -> permite ler uma string com espaços palavra por palavra.

Exemplo: string nome = Lucas Santos;
stringstream pegar(nome);
string palavra;
pegar palavra(funciona igual o cin)
palavra agora é igual a Lucas, com loop da pra pegar outras palavras.