A photograph of a green ceramic cup filled with a latte, featuring a white leaf-shaped latte art design. The cup sits on a matching green saucer, which is placed on a rustic wooden table. The background is softly blurred, showing more of the table and a hint of a bright, possibly windowed area.

Boas práticas de desenvolvimento de software (continuação)

Aula prática

Aula anterior



Sorteando números

Número aleatório em C++

```
#include <iostream> // Para cout  
#include <cstdlib> // Para rand()  
#include <ctime> // Para time()
```

```
int main() {
```

```
    int X = 10;
```

```
    cout << endl << X;
```

```
    return 0;
```

```
} // fim main()
```

Número aleatório em C++

```
#include <iostream> // Para cout  
#include <cstdlib> // Para rand()  
#include <ctime> // Para time()
```

```
int main() {
```

```
    int X = rand ();
```

```
    cout << endl << X;
```

```
    return 0;
```

```
} // fim main()
```

Número aleatório em C++

```
#include <iostream> // Para cout
#include <cstdlib> // Para rand()
#include <ctime> // Para time()

int main() {

    int X = rand () % 11; // Limita a intervalo entre 0 e 10

    cout << endl << X;

    return 0;
} // fim main()
```

Número aleatório em C++

```
#include <iostream> // Para cout
#include <cstdlib> // Para rand()
#include <ctime> // Para time()

int main() {
    srand( time(0) ); //inicializa a semente com o tempo atual
    int X = rand () % 11; // Limita a intervalo entre 0 e 10

    cout << endl << X;

    return 0;
} // fim main()
```

Tente

1. Um vetor deve armazenar uma coleção de 10 números. Os números precisam estar em um intervalo fechado de 0 a 100. Sorteie os números para o vetor e o escreva.
2. Construa um programa que sugira seis números para o jogo da Megasena.
3. Joguinho: Lançar um dado e apresentar o valor [1..6]

O laço *for-each*

O laço *for-each*

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
```

```
    string array[]={"Lucas", "Gabriel", "Any", "Mateus"};
```

```
    for( string nome : array) {
```

```
        cout << endl << nome;
```

```
    } // fim for(nome)
```

```
    return 0;
```

```
} // fim main()
```

O laço *for-each*

```
class Aluno {  
    private :  
        string nome;  
    public :  
        Aluno(){};  
        Aluno(string nome){  
            setNome(nome);  
        }  
        string getNome(){  
            return this->nome;  
        }  
        void setNome(string nome){  
            this->nome = nome;  
        }  
};
```


```
#include <iostream>  
using namespace std;  
  
int main() {  
    Aluno* array[3];  
    array[0] = new Aluno("Lucas");  
    array[1] = new Aluno("Gabriel");  
    array[2] = new Aluno("Any");  
  
    // Listando toda a turma  
    for( Aluno* aluno : array) {  
        cout << aluno->getNome() << endl;  
    }  
    return 0;  
} // fim main()
```

Tratamento de Exceções

Exemplo: Operação em objeto NULL

```
class Aluno {  
    private :  
        string nome;  
    public :  
        Aluno(){};  
        Aluno(string nome){  
            setNome(nome);  
        }  
        string getNome(){  
            return this->nome;  
        }  
        void setNome(string nome){  
            this->nome = nome;  
        }  
};
```

```
#include <iostream>  
using namespace std;  
  
int main() {  
    Aluno* array[3];  
    array[0] = new Aluno("Lucas");  
    array[1] = NULL;  
    array[2] = new Aluno("Any");  
  
    // Listando toda a turma  
    for( Aluno* aluno : array) {  
        cout << aluno->getNome() << endl;  
    }  
    return 0;  
} // fim main()
```

A photograph of a green ceramic cup filled with a latte, featuring a white leaf-shaped latte art design. The cup sits on a matching green saucer, which is placed on a rustic wooden table. The background is softly blurred, showing more of the wooden surface and a hint of a bright, out-of-focus light source.

Boas práticas de desenvolvimento de software

Tratamento de Exceções (continuação)

Alternativa:

Mecanismo para tratamento de exceções

Permite tratar uma exceção antes de abortar a execução do código.
Para isto, ao gerar um erro, uma exceção é lançada.

Palavras-chave:

try

throw

catch

Formam a instrução **try..catch()**

```
#include <stdexcept> // arquivo de cabeçalho stdexcept contém runtime_error  
using std::runtime_error; // classe runtime_error da biblioteca-padrão do C++
```


Lembre-se das bolas nos quintais das casas...



Implementação

A instrução
`try..catch`

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {

    }
    catch ( )
    {

    }

    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        throw

    }
    catch ( )
    {

    }

    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        cout << "\nEntrei no bloco Try" << endl;
    }
    catch ( )
    {

    }

    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        cout << "\nEntrei no bloco Try" << endl;
    }
    catch (...)
    {

    }

    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        cout << "\nEntrei no bloco Try" << endl;
    }
    catch (...)
    {

    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        cout << "\nEntrei no bloco Try" << endl;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        cout << "\nEntrei no bloco Try" << endl;
        throw ;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```



```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        cout << "\nEntrei no bloco Try" << endl;
        throw 1;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        cout << "\nEntrei no bloco Try" << endl;
        throw 1;
    }
    catch (...) // Há algum problema aqui?
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        cout << "\nEntrei no bloco Try" << endl;
        throw 1;
    }
    catch (...) // Único catch a ser executado
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        cout << "\nEntrei no bloco Try" << endl;
        throw 1;
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro" << endl;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        int a = 10;
        cout << "\nEntrei no bloco Try" << endl;
        throw 1;
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro com a = " << a << endl; // Erro???
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    int a;
    try
    {
        a = 10;
        cout << "\nEntrei no bloco Try" << endl;
        throw a;
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro com a = " << a << endl;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    try
    {
        int a = 10;
        cout << "\nEntrei no bloco Try" << endl;
        throw a;
    }
    catch (int x) // Variável paramétrica em catch
    {
        cout << "\nEntrei no bloco Catch de inteiro com x = " << x << endl;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    int a;
    try
    {
        a = 10;
        cout << "\nEntrei no bloco Try" << endl;
        throw 7.0; // Qual catch será executado?
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro com x = " << x << endl;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```



```
int main()
{
    cout << "Hello world!" << endl;
    int a;
    try
    {
        a = 10;
        cout << "\nEntrei no bloco Try" << endl;
        throw 7.0; // Qual catch será executado?
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro com x = " << x << endl;
    }
    catch (double x)
    {
        cout << "\nEntrei no bloco Catch de real com x= " << x << endl;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    int a;
    try
    {
        a = 10;
        cout << "\nEntrei no bloco Try" << endl;
        throw 7.0;
        cout << "\nDory disse: Me escolhe, me escolhe!" << endl; // Será executada?
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro com x = " << x << endl;
    }
    catch (double x)
    {
        cout << "\nEntrei no bloco Catch de realcom x= " << x << endl;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    int a;
    try
    {
        a = 10;
        cout << "\nEntrei no bloco Try" << endl;
        if(a > 10) throw 7.0;
        cout << "\nDory disse: Me escolhe, me escolhe!" << endl; // Será executada?
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro com x = " << x << endl;
    }
    catch (double x)
    {
        cout << "\nEntrei no bloco Catch de real com x= " << x << endl;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    int a;
    try
    {
        a = 10;
        cout << "\nEntrei no bloco Try" << endl;
        if(a == 10) throw 7.0;
        cout << "\nDory disse: Me escolhe, me escolhe!" << endl; // Será executada?
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro com x = " << x << endl;
    }
    catch (double x)
    {
        cout << "\nEntrei no bloco Catch de real com x = " << x << endl;
    }
    catch (...)
    {
        cout << "\nEntrei no bloco Catch padrao" << endl;
    }

    cout << "\nSai da instrucao Try" << endl;
    return 0;
}
```

```
int main()
{
    cout << "Hello world!" << endl;
    int a;
    try
    {
        a = 10;
        cout << "\nEntrei no bloco Try" << endl;
        if(a == 10) throw 'a';
        cout << "\nDory disse: Me escolhe, me escolhe!" << endl;
    }
    catch (int x)
    {
        cout << "\nEntrei no bloco Catch de inteiro com x = " << x << endl;
    }
    catch (double x)
    {
        cout << "\nEntrei no bloco Catch de real com x= " << x << endl;
    }

    cout << "\n Esta msg serah executada???" << endl;
    return 0;
}
```

```
int main() {  
    int a, b;  
    cout << "\nDivisao inteira. Digite os dois operandos: ";  
    cin >> a >> b;  
    cout << "\nResultado: " << a / b;  
    cout << "\nNunca executado quando divisor igual a zero – Execucao abortada";  
    return 0;  
}
```

```
int main()
{
    int a, b;
    try
    {
        cout << "\nDivisao inteira. Digite os dois operandos: ";
        cin >> a >> b;
        if(b==0) throw runtime_error("\n\taErro: Divisao por zero");
        cout << "\nResultado: " << a / b;
    }
    catch(runtime_error e)
    {
        cout << "\nDivisao por zero\n";
    }
    cout << "\nDivisao não executada quando divisor igual a zero";
    return 0;
}
```

```
int main()
{
    int a, b;
    try
    {
        cout << "\nDivisao inteira. Digite os dois operandos: ";
        cin >> a >> b;
        if(b==0) throw runtime_error("\n\taErro: Divisao por zero");
        cout << "\nResultado: " << a / b;
    }
    catch(runtime_error e)
    {
        cout << e.what() << endl;
    }
    cout << "\nDivisao não executada quando divisor igual a zero";
    return 0;
}
```



```
int divisao(int x, int y){  
    int resultado;  
    try{  
        if(y == 0) throw runtime_error("Erro: divisao por zero");  
        else resultado= x / y;  
    }  
    catch(runtime_error e){  
        resultado = 0;  
        cout << e.what() << endl;  
    }  
    return resultado;  
}
```

```
int main() {  
    int a, b;  
    cout << "\nDivisao inteira. Digite os dois operandos: ";  
    cin >> a >> b;  
    cout << "\nResultado: " << divisao(a,b);  
    cout << "\nSempre executado, ainda que divisor igual a zero";  
    return 0;  
}
```

```
int divisao(int x, int y){  
    if(y == 0) throw runtime_error("Erro: divisao por zero");  
    else return x / y;  
}
```

```
int main() {  
    int a, b;  
    cout << "\nDivisao inteira. Digite os dois operandos: ";  
    cin >> a >> b;  
  
    try{  
        cout << "\nResultado: " << divisao(a,b);  
    }  
    catch(runtime_error e){  
        cout << e.what();  
    }  
  
    cout << "\nSempre executado, ainda que divisor igual a zero";  
    return 0;  
}
```

```
int divisao(int x, int y){
    if(y == 0) throw runtime_error("Erro: divisao por zero");
    else return x / y;
}

int main() {
    int a, b;
    bool execucao=true;

    do{
        cout << "\nDivisao inteira. Digite os dois operandos: ";
        cin >> a >> b;
        try{
            cout << "\nResultado: " << divisao(a,b);
            execucao=false;
        }
        catch(runtime_error e){
            cout << e.what();
        }
    }while(execucao);

    cout << "\nSempre executado, ainda que divisor igual a zero";
    return 0;
}
```

```
#include <iostream>
using namespace std;
```

```
int main()
```

```
{
```

```
    double *ptr[ 200 ];
```

```
    // aloca memória para ptr
```

```
    for ( int i = 0; i < 200; i++ )
```

```
    {
```

```
        ptr[ i ] = new double[ 50000000 ]; // pode lançar exceção
```

```
        cout << "Allocated 50.000.000 doubles in ptr[ " << i << " ]\n";
```

```
    } // fim do for
```

```
} // fim de main
```

```
#include <iostream>
using namespace std;
```

```
#include <new> // operador new padrão
using std::bad_alloc;
```

```
int main()
{
    double *ptr[ 200 ];

    // aloca memória para ptr
    try
    {
        // aloca memória para ptr[i]; new lança bad_alloc em caso de falha
        for ( int i = 0; i < 200; i++ )
        {
            ptr[ i ] = new double[ 50000000 ]; // pode lançar exceção
            cout << "Allocated 50000000 doubles in ptr[ " << i << " ]\n";
        } // fim do for
    } // fim do try
}
```

```
#include <iostream>
using namespace std;
#include <new> // operador new padrão
using std::bad_alloc; // classe bad_alloc de new, relativa à falha ao tentar alocar memória
int main() {
    double *ptr[ 200 ];
    try {
        // aloca memória para ptr[i]; new lança bad_alloc em caso de falha
        for ( int i = 0; i < 200; i++ ) {
            ptr[ i ] = new double[ 50000000 ]; // pode lançar exceção
            cout << "Allocated 50000000 doubles in ptr[ " << i << " ]\n";
        } // fim do for
    } // fim do try

    // trata exceção bad_alloc
    catch ( bad_alloc memoryAllocationException ) {
        cout << "Exception occurred: " << memoryAllocationException.what() << endl;
        cout << "\nMemoria insuficiente";
    } // fim do catch

    return 0;
} // fim de main
```

```
int main() {
    double *ptr[ 200 ];
    int i;
    try {
        for ( i = 0; i < 200; i++ ) {
            ptr[ i ] = new double[ 50000000 ]; // pode lançar exceção
            cout << "Allocated 50000000 doubles in ptr[ " << i << " ]\n";
        } // fim do for
    } // fim do try

    // trata exceção bad_alloc
    catch ( bad_alloc memoryAllocationException ) {
        cout << "\nMemoria insuficiente. Continuando com apenas " << i << " elementos.";
    } // fim do catch

    // A execução não interrompe independentemente do número de elementos alocados
    for ( int x = 0; x < i; x++ ) {
        for( int j = 0; j < 50000000 ; j++ ) { // Reduza para 500 para viabilizar o tempo de execução
            ptr[x][j] = x+1;
            cout << "\nVisitando " << ptr[x][j];
        } // fim do for j
    } // fim do for x

    return 0;
} // fim de main
```

Exemplo: Operação em objeto NULL

```
#include <iostream>
#include <stdexcept>
using namespace std;

class Objeto {
public:
    void mostrarMensagem() {
        cout << "Objeto inicializado e funcional!" << endl;
    }
};

void processarObjeto(Objeto* obj) {
    if (obj == nullptr) {
        throw runtime_error("Erro: Objeto é NULL.");
    }
    obj->mostrarMensagem();
}
```

```
int main() {
    Objeto* meuObjeto = nullptr; // Objeto não inicializado
    try {
        processarObjeto(meuObjeto);
    } catch (const runtime_error& e) {
        cout << e.what() << endl;
    }

    // Corrigindo o problema
    cout << "Inicializando objeto..." << endl;
    meuObjeto = new Objeto();
    try {
        processarObjeto(meuObjeto);
    } catch (const runtime_error& e) {
        cout << e.what() << endl;
    }

    delete meuObjeto; // Libera a memória alocada
    return 0;
}
```


Trabalho Prático

Acrescente ao seu trabalho o Tratamento de Exceções

Em caso de falha ao criar novos arquivos ou novas instâncias de Pessoa - Aluno ou Professor, comunique ao usuário ter atingido o limite dos recursos e faça o fluxo de execução voltar ao menu principal.