

ECOP13A-Lab9

Tratamento de Exceções

Guia de Laboratório
Prof. André Bernardi
andrebernardi@unifei.edu.br





9º Laboratório ECOP13A

13 junho 2025



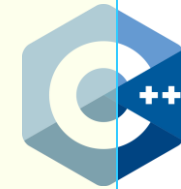
1ª Questão



Utilizar a classe polinômio do laboratório 6 (Ex1) para implementar o lançamento de uma exceção de índice fora da faixa, dentro da sobrecarga do operador `[]`.

Criar um programa para testar o mecanismo das exceções, utilizando esta classe.

1ª questão – Exemplo de Solução



```
#ifndef POLINOMIO_H
#define POLINOMIO_H

#include <iostream>
using namespace std;

class Polinomio{
private:
    double *valores;
    int n;

public:
    Polinomio();
    Polinomio(int);
    Polinomio(const Polinomio&);

    virtual ~Polinomio();

    Polinomio operator = ( const Polinomio& );
    Polinomio operator+(Polinomio);
    Polinomio operator-(Polinomio);

    double& operator[](int); // lança uma exceção out of range

    friend ostream& operator << (ostream&, Polinomio&);
    friend istream& operator >> (istream&, Polinomio&);
};
#endif
```

```
#include <iostream>
#include "polinomio.h"
#include <stdexcept> // EX1 - LAB8
```

```
using namespace std;
```

```
Polinomio::Polinomio()
```

```
{
    n = 2;
    valores = new double[n];
    valores[0] = 1;
    valores[1] = 1;
}
```

```
Polinomio::Polinomio(int _n)
```

```
{
    n = _n + 1;
    valores = new double[n];
    for(int i = 0; i < n; i++)
        valores[i] = 1;
}
```

```
//construtor de copia é necessário pois a classe usa ptr
```

```
Polinomio::Polinomio(const Polinomio& p)
```

```
{
    n = p.n;
    valores = new double[n];
    for(int i = 0; i < p.n; i++)
    {
        valores[i] = p.valores[i];
    }
}
```



```
Polinomio::~~Polinomio()
```

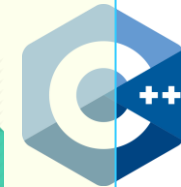
```
{  
    delete[] valores;  
}
```

```
Polinomio Polinomio::operator=(const Polinomio& p)
```

```
{  
    delete [] valoers;          // limpar o poonteiro antigo  
    n = p.n;  
    valores = new double[n];    // alocar para o novo tamanho  
    for(int i = 0; i < p.n; i++)  
    {  
        valores[i] = p.valores[i]; //copiar valores  
    }  
    return *this;  
}
```

```
Polinomio Polinomio::operator+(Polinomio _pol)
```

```
{  
    Polinomio temp(max(_pol.n, n)-1);  
    int i;  
    for(i = 0; i < min(_pol.n, n); i++)  
        temp[i] = _pol.valores[i] + valores[i];  
  
    if(_pol.n > n)  
        for(int j = i; j < _pol.n; j++)  
            temp[j] = _pol.valores[j];  
    else  
        for(int j = i; j < n; j++)  
            temp[j] = valores[j];  
    return temp;  
}
```



```
Polinomio Polinomio::operator-(Polinomio _pol)
```

```
{
```

```
    Polinomio temp(max(_pol.n, n)-1);
```

```
    int i;
```

```
    for(i = 0; i < min(_pol.n, n); i++)  
        temp[i] = valores[i] - _pol.valores[i];
```

```
    if(_pol.n > n)  
        for(int j = i; j < _pol.n; j++)  
            temp[j] = - _pol.valores[j];
```

```
    else
```

```
        for(int j = i; j < n; j++)  
            temp[j] = valores[j];
```

```
    return temp;
```

```
}
```

```
istream& operator >> (istream& input, Polinomio& _pol)
```

```
{
```

```
    cout << "Polinomio: C0 + C1x1 + C2x2 + ... + Cnxn = 0" << endl;
```

```
    for(int i = 0; i < _pol.n; i++)
```

```
    {
```

```
        cout << "Digite o valor de C" << i << ": ";
```

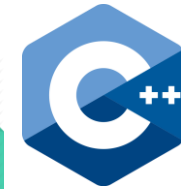
```
        input >> _pol[i];
```

```
    }
```

```
    return input;
```

```
}
```





```
ostream& operator << (ostream& output, Polinomio& _pol)
{
    for(int i = 0; i < _pol.n; i++){
        if(i != _pol.n-1)
            output << _pol[i] << "x^" << i << ((_pol[i+1]>=0)? " + " : " ");
        else
            output << _pol[i] << "x^" << i << " = 0";
    }
    return output;
}

//alterada para EX1 - LAB8
double& Polinomio::operator[](int pos)
{
    if(pos >= 0 && pos < n)
        return valores[pos];
    else
        throw out_of_range("Indice Invalido!!");//return valores[0];
}
```



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

```
int main(){
    Polinomio a(3), b(4), c(4);
```

```
    cin >> a >> b;
    cout << a << endl;
    cout << b << endl;
```

```
    c = b - a;
    cout << c << endl;
    c = a + b;
    cout << c << endl;
```

```
    int index;
    cout << " Entre com um indice: ";
    cin >> index;
```

```
    try{
        //alterar o elemento usando operador []
        //c[index] = 10;
        cout << "O coeficiente do grau " << index << " vale: "
              << c[index] << endl;
```

```
    }
```

```
    catch(out_of_range &ex)
```

```
    {
```

```
        cout << ex.what() << " nao e possivel mostrar o coeficiente." << endl;
```

```
    }
```

```
    return 0;
```

```
}
```

Main





"D:\2022\Aulas\ecop13A\9 - Tratamento de Excepções\Lab8\Ex1\bin\Debug\Ex1.exe"

Polinomio: $C_0 + C_1x^1 + C_2x^2 + \dots + C_nx^n = 0$

Digite o valor de C_0 : 1

Digite o valor de C_1 : 1

Digite o valor de C_2 : 1

Digite o valor de C_3 : 1

Polinomio: $C_0 + C_1x^1 + C_2x^2 + \dots + C_nx^n = 0$

Digite o valor de C_0 : 2

Digite o valor de C_1 : 2

Digite o valor de C_2 : 2

Digite o valor de C_3 : 2

Digite o valor de C_4 : 2

$1x^0 + 1x^1 + 1x^2 + 1x^3 = 0$

$2x^0 + 2x^1 + 2x^2 + 2x^3 + 2x^4 = 0$

$-1x^0 -1x^1 -1x^2 -1x^3 -2x^4 = 0$

$3x^0 + 3x^1 + 3x^2 + 3x^3 + 2x^4 = 0$

Entre com um indice: 5

o coeficiente do grau 5 vale: Indice Invalido!! nao e possivel mostrar o coeficiente.

Process returned 0 (0x0) execution time : 21.120 s

Press any key to continue.

2ª Questão



Crie um programa que aloque continuamente um vetor de double, sem desalocá-lo para testar o processamento de exceções de falta de memória. (`std::bad_alloc`).

Observar o monitor de recursos do sistema para acompanhar a alocação de memória.



2ª Exemplo de Solução 1

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main(){
6     double *p;
7     try{
8         while(1){
9             p = new double[10];
10            p[0] = 1;
11        }
12    }catch(bad_alloc &e){
13        cout << e.what() << endl;
14    }
15 }
```

Cuidado ao executar!
Pode travar seu sistema.

Para gerar a exceção `bad_alloc` com segurança, sem travar seu sistema, substitua a linha 9 por:

`p = new double[1000000000000];`



2ª Exemplo de Solução 2

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

struct No {
    double value;
    No *prox;
    No () {
        value = 1.f;
    }
};

int main() {

    try {
        No *v = new No;

        for (;;) {
            No *novo = new No;
            novo->prox = v;
            v = novo;
        }
    } catch (bad_alloc b) {
        cout << b.what() << endl;
    }
}
```

Cuidado ao executar!
Pode travar seu sistema.

3ª Questão



Para a classe CPilha abaixo, identifique e corrija os erros de sintaxe presentes em sua descrição.

Sobrecarregar os métodos de impressão e leitura para este objeto.

3ª questão



```
//arquivo vetor.h
//header file para classe pilha
#ifndef PILHA_H
#define PILHA_H

class CPilha
{
    Private
    int *m_dados;           // ponteiro para dados da pilha
    int m_ponteirodaPilha   // apontador do topo da pilha
    int m_tamanho;          // espaço de memória reservado para o objeto

    public
    void CPilha ( memoria );// construtor com parâmetros
    CPilha ( void );        // construtor sem parâmetros
    ~CPilha ( void );       // destrutor

    int pop ( int *pop_to ); // puxar dados da pilha
    int push ( int push_this );// empurrar dados para a pilha
}
#endif
```



3ª questão – Exemplo de solução



Erros de sintaxe:

```
//arquivo vetor.h
//header file para classe pilha
#ifndef PILHA_H
#define PILHA_H

class CPilha
{
    Private
    int *m_dados;           // ponteiro para dados da pilha
    int m_ponteirodaPilha  // apontador do topo da pilha
    int m_tamanho;         // espaço de memória reservado para o objeto

    public
    void CPilha ( memoria ); // construtor com parâmetros
    CPilha ( void );         // construtor sem parâmetros
    ~CPilha ( void );        // destrutor

    int pop ( int *pop_to ); // puxar dados da pilha
    int push ( int push_this ); // empurrar dados para a pilha
}
#endif
```

Palavra reservada – **private**

Bloco de acesso deve terminar em :

Falta ponto e virgula ;

Bloco de acesso deve terminar em :

Construtor **não** tem tipo de retorno.

Parâmetro em C++ deve ser um **tipo**.

Falta ponto e virgula ;



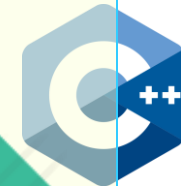

```

1 //header file para classe pilha
2 #ifndef PILHA_H
3 #define PILHA_H
4
5 #include <iostream>
6 using namespace std;
7
8 class CPilha
9 {
10     private:
11         int *m_dados;           // ponteiro para dados da pilha
12         int m_ponteirodaPilha;  // apontador do topo da pilha
13         int m_tamanho;          // espaço de memória reservado para o objeto
14
15     public:
16         CPilha ( int memoria = 0 ); // construtor com parâmetros
17         // CPilha ( void );          // construtor sem parâmetros
18         virtual ~CPilha ( void );  // destrutor
19
20         int pop ( int &pop_to );    // puxar dados da pilha
21         int push ( int push_this ); // empurrar dados para a pilha
22
23         CPilha ( const CPilha& );   // construtor de cópia
24         CPilha& operator = (const CPilha&); // operador de atribuição
25
26         friend ostream& operator << ( ostream& , const CPilha& );
27         friend istream& operator >> ( istream& , CPilha& );
28 };
29 #endif

```

3ª questão

Exemplo de Solução:



4ª Questão



Implemente a classe **CPilha** do exercício anterior acrescentando o lançamento de exceções nas funções **push** e **pop** para sinalizar que não conseguiu inserir ou retirar um elemento de dentro da pilha, respectivamente.

Implemente um programa principal simples para testar suas funcionalidades.

4ª questão:



```
#include "cpilha.h"
#include <stdexcept>

using namespace std;

ostream& operator <<(ostream& out, const CPilha& p)
{
    for (int i = 0; i <= p.m_ponteirodaPilha; i++)
        out << p.m_dados[i] << " ";
    return out;
}

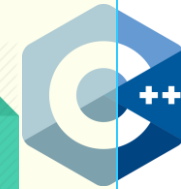
istream& operator >>(istream& in, CPilha& p)
{
    int a;
    p.m_ponteirodaPilha = -1; // esvaziar a pilha
    cout << "Entre com os dados da pilha: ";
    try{
        for (int i = 0; i < p.m_tamanho ; i++)
        {
            in >> a;
            p.push(a);
        }
    }
    catch (runtime_error & rt){
        cout << rt.what() << endl;
    }
    return in;
}
```

```
// construtor com parâmetros
CPilha::CPilha ( int memoria )
{
    m_ponteirodaPilha = -1;    // sinaliza pilha vazia
    m_tamanho = (memoria > 0)? memoria : 10;
    m_dados = new int [ m_tamanho ];
}

// destrutor
CPilha::~CPilha ( void )
{
    delete [] m_dados;
}

// puxar dados da pilha
int CPilha::pop ( int &pop_to )
{
    if (m_ponteirodaPilha == -1)
        throw runtime_error("Pilha esta vazia!!!");
    pop_to = m_dados[m_ponteirodaPilha--];
    return pop_to;
}

// empurrar dados para a pilha
int CPilha::push ( int push_this )
{
    if (m_ponteirodaPilha == m_tamanho-1)
        throw runtime_error("Pilha cheia!!");
    m_dados[++m_ponteirodaPilha] = push_this;
    return push_this;
}
```





```
// construtor de cópia
CPilha::CPilha ( const CPilha& p )
{
    m_ponteirodaPilha = p.m_ponteirodaPilha;
    m_tamanho = p.m_tamanho;
    m_dados = new int [ m_tamanho ];
    for (int i = 0; i <= m_ponteirodaPilha ; i++)
        m_dados[i] = p.m_dados[i];
}
```

```
// operador de atribuição
CPilha& CPilha::operator = (const CPilha& p)
{
    delete [] m_dados;
    m_ponteirodaPilha = p.m_ponteirodaPilha;
    m_tamanho = p.m_tamanho;
    m_dados = new int [ m_tamanho ];
    for (int i = 0; i <= m_ponteirodaPilha ; i++)
        m_dados[i] = p.m_dados[i];

    return *this;
}
```

main

CPP



```
#include <iostream>
#include "cpilha.h"

using namespace std;

int main()
{
    CPilha pInt;
    int aux;
    try
    {
        // usando a pilha de inteiros
        cin >> pInt;
        cout << "Valor no topo da pilha: " << pInt.pop(aux) << endl;
        cout << pInt;
    }
    catch (runtime_error & rt)
    {
        cout << rt.what() << endl;
    }

    return 0;
}
```