Fatec Baixada Santista Rubens Lara

Aluna: Mylla Cristhina Rodrigues da Costa Abreu Korres RA: 0050482011032

Curso: Análise e Desenvolvimento de Sistema Manhã Período: Matutino

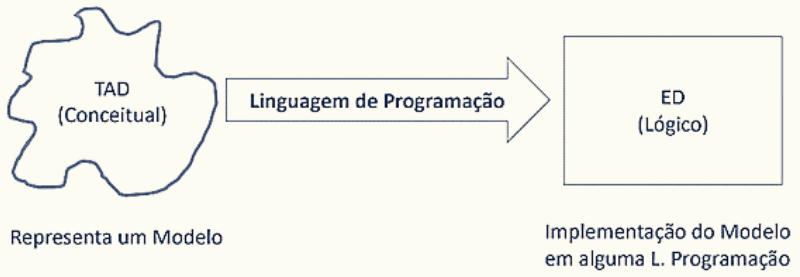
Disciplina: Estrutura de Dados Professor: Chiara

TP5 – Lista de Exercícios P1

1. O que podemos entender por um tipo abstrato de dados? E uma estrutura de dados?

Tipo abstrato de dados (TAD) é a representação de um conjunto de dados e um conjunto de procedimentos (funções) que podem ser aplicadas sobre esses dados.

Já uma estrutura de dados (ED) é uma implementação de um tipo abstrato de dados. Ela representa as relações lógicas entre os dados.



2. O que são listas lineares?

São estruturas de dados onde elementos de um mesmo tipo de dado estão organizados de maneira sequencial, ou seja, existe uma ordem lógica entre eles.

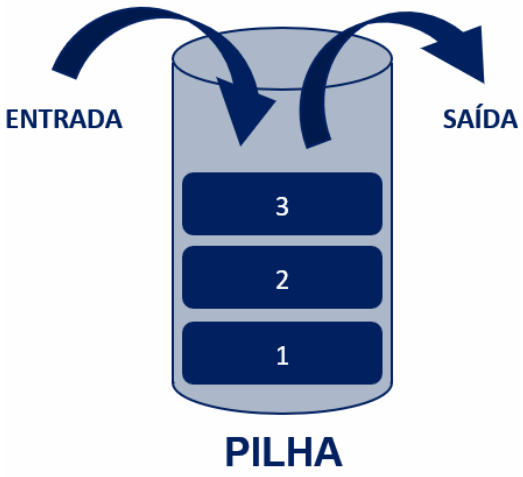
É uma sequência de n elementos, tal que:

- n é a quantidade de elementos da lista;
- Se n=0, então a lista está vazia;
- x_1 é o primeiro elemento da lista;
- x_n é o último elemento da lista;
- Para 1< k < n, o elemento x_k tem x_{k-1} como seu antecessor e x_{k+1} como seu sucessor.

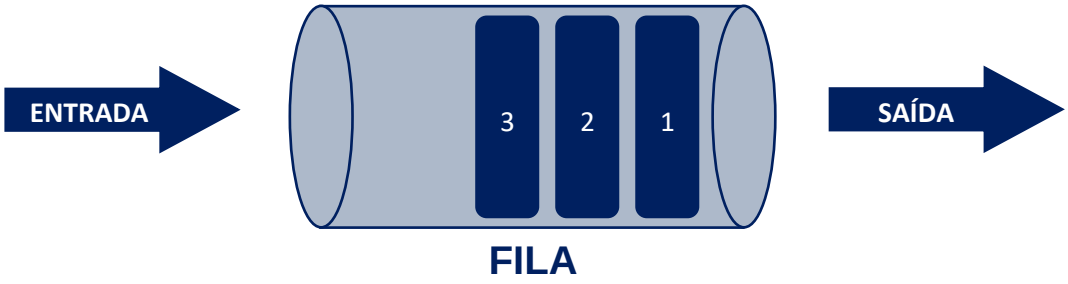
3. O que são listas lineares restritas? Cite 2 tipos e respectivas definições.

São listas lineares que possuem uma limitação para poderem representar alguns modelos do mundo real. As mais utilizadas são:

- **Pilhas**: lista linear onde tanto a entrada quanto a saída são feitas pela mesma extremidade. Ou seja, o último elemento a entrar será o primeiro a sair (Last In / First Out);



- **Filas**: lista linear onde a entrada é feita por uma extremidade e a saída é feita por outra. Ou seja, o primeiro elemento a entrar é o primeiro a sair (First In / First Out).



FILA

4. Considere os protótipos, definidos abaixo, para uma estrutura de dados do tipo fila, implementada em alocação sequencial com representação circular.

void insere(int x): //insere um novo elemento na fila
int remove(): //remove um elemento da fila, retornando este elemento removido
int primeiro(): //retorna o valor do primeiro elemento da fila.

Mostre a situação de uma fila, inicialmente vazia, após cada uma das seguintes operações:

1. insere(5)	2. insere(8)	3. insere(2)
4. insere(primeiro())	5. remove()	6. insere(remove())
7. remove()	8. insere(1)	9. remove()
10. insere(remove())	11. insere(primeiro())	12. insere(remove())
13. insere(3)	14. insere(remove())	15. insere(3)

1. insere(5)
Fila: 5

2. insere(8)
Fila: 5 8

3. insere(2)
Fila: 5 8 2

4. insere(primeiro())
Fila: 5 8 2 5

5. remove()
Fila: 8 2 5

6. insere(remove())
Fila: 2 5 8

7. remove()
Fila: 5 8

8. insere(1)
Fila: 5 8 1

9. remove()
Fila: 8 1

10. insere(remove())
Fila: 1 8

11. insere(primeiro())
Fila: 1 8 1

12. insere(remove())
Fila: 8 1 1

13. insere(3)
Fila: 8 1 1 3

14. insere(remove())
Fila: 1 1 3 8

15. insere(3)
Fila: 1 1 3 8 3

5. Considere os protótipos, definidos abaixo, para uma estrutura de dados do tipo Pilha, implementada em alocação sequencial:

a) void empilha(int x): - empilha um novo elemento
b) int desempilha(): - desempilha um elemento e retorna o elemento desempilhado
c) int topo(): - retorna o valor do elemento do topo da pilha

Mostre a situação de uma pilha , inicialmente vazia, após a execução de cada umas das operações:

1. empilha(1)	2. empilha(5)	3. desempilha()
4. empilha(5)	5. empilha(9)	6. empilha(7)
7. empilha(topo())	8. desempilha()	9. empilha(desempilha())
10. empilha(4)	11. empilha(3)	12. empilha(8)

1. empilha(1)	**5. empilha(9)**	**9. empilha(desempilha())**
Pilha: 1	Pilha: 1 5 9	Pilha: 1 5 9 7
2. empilha(5)	**6. empilha(7)**	**10. empilha(4)**
Pilha: 1 5	Pilha: 1 5 9 7	Pilha: 1 5 9 7 4
3. desempilha()	**7. empilha(topo())**	**11. empilha(3)**
Pilha: 1	Pilha: 1 5 9 7 7	Pilha: 1 5 9 7 4 3
4. empilha(5)	**8. desempilha ()**	**12. empilha(8)**
Pilha: 1 5	Pilha: 1 5 9 7	Pilha: 1 5 9 7 4 3 8

6. Utilizando as implementações de uma estrutura de dados pilha, elabore um programa que converta um número decimal em binário.

```
#include<iostream>
#include<iomanip>
#include<locale>
#include<stdlib.h>
#include<stdio.h>
#include<windows.h>
#include"pilha.h"

using namespace std;

HANDLE color = GetStdHandle(STD_OUTPUT_HANDLE);

Pilha<int>p(16);

void bin(int dec)
{
	while (dec != 0)
	{
		if (dec % 2 == 0)
		{
			p.empilha(0);
		}
		else
		{
			p.empilha(1);
		}

		dec /= 2;
	}
}

int main()
{
	setlocale(LC_ALL, "Portuguese");

	int dec;
	char c;

	do
	{
		system("cls");

		do
		{
```

```
                cout << "\n\n\t\t Insira um número decimal: ";
                cin >> dec;
        } while (dec < 0);

        system("cls");

        cout << "\n\n\n\n\t\t DECIMAL: " << dec;

        bin(dec);

        cout << "\n\n\n\t\t BINÁRIO: ";

        SetConsoleTextAttribute(color, 11);

        if (!p.pilhavazia())
        {
                for (int i = p.getTopo(); i >= 0; i--)
                {
                        cout << p.getValor(i);
                }
        }
        else
        {
                cout << "0";
        }

        SetConsoleTextAttribute(color, 15);

        cout << "\n\n\n\n\t\t DESEJA CONVERTER OUTRO DECIMAL EM BINÁRIO? S/N ";
        cin >> c;

        if (c != 's' && c != 'S')
        {
                system("cls");

                SetConsoleTextAttribute(color, 15);

                cout << "\n\n\n\n\n\t\t\t FINALIZANDO..." << endl;

                Sleep(2000);

                system("cls");

                SetConsoleTextAttribute(color, 10);

                cout << "\n\n\n\n\n\t\t\t >>>>> PROGRAMA FINALIZADO <<<<<\n\n" << endl;

                SetConsoleTextAttribute(color, 15);
        }
        else
        {
                while (!p.pilhavazia())
                {
                        p.desempilha();
                }
        }

} while (c == 's' || c == 'S');

return 0;
}
```

7. Considerando uma pilha e uma fila, de números inteiros, construa uma aplicação que:

a) define uma pilha e uma fila de tamanho 20.
b) carrega a fila com números aleatórios compreendidos entre 35 e 78
b) exibe a fila
c) transfere todos os elementos da fila para a pilha
d) exibe a pilha

```
#include<iostream>
#include<iomanip>
#include<locale>
#include<stdlib.h>
#include<stdio.h>
#include<windows.h>
#include<time.h>
#include"fila.h"
#include"pilha.h"

using namespace std;

int main()
{
	HANDLE color = GetStdHandle(STD_OUTPUT_HANDLE);

	setlocale(LC_ALL, "Portuguese");

	char c;
	Fila<int>f(20);  // a)
	Pilha<int>p(20); // a)

	do
	{
		system("cls");

		srand(time(NULL));

		for (int i = 0; i < 20; i++)
		{
			f.insere(35 + rand() % 44); // b)
		}

		SetConsoleTextAttribute(color, 15);

		cout << "\n\n\t\t FILA: ";

		SetConsoleTextAttribute(color, 11);

		if (f.getInic() > f.getFim()) // c)
		{
			for (int i = f.getInic(); i < f.getTamanho(); i++)
			{
				cout << f.getValor(i) << " ";
			}

			for (int i = 0; i <= f.getFim(); i++)
			{
				cout << f.getValor(i) << " ";
			}
		}
		else
		{
```

```
                for (int i = f.getInic(); i <= f.getFim(); i++)
                {
                        cout << f.getValor(i) << " ";
                }
        }

        SetConsoleTextAttribute(color, 15);

        cout << "\n\n\n\t\t TRANSFERINDO OS ELEMENTOS..." << endl;

        SetConsoleTextAttribute(color, 15);

        Sleep(2000);

        while (!f.filavazia()) // d)
        {
                p.empilha(f.remove());
        }

        cout << "\n\n\t\t PILHA: ";

        SetConsoleTextAttribute(color, 14);

        for (int i = p.getTopo(); i >= 0; i--) // e)
        {
                cout << p.getValor(i) << " ";
        }

        SetConsoleTextAttribute(color, 15);

        cout << "\n\n\n\n\t\t DESEJA CRIAR OUTRA FILA DE NÚMEROS ALEATÓRIOS? S/N ";
        cin >> c;

        if (c != 's' && c != 'S')
        {
                system("cls");

                cout << "\n\n\n\n\n\t\t\t FINALIZANDO..." << endl;

                Sleep(2000);

                system("cls");

                SetConsoleTextAttribute(color, 10);

                cout << "\n\n\n\n\n\t\t\t >>>>> PROGRAMA FINALIZADO <<<<<\n\n" << endl;

                SetConsoleTextAttribute(color, 15);
        }
        else
        {
                while (!p.pilhavazia())
                {
                        p.desempilha();
                }
        }

} while (c == 's' || c == 'S');

return 0;
}
```

8. Um pangrama é uma frase que contém pelo menos uma vez cada uma das 26 letras do novo alfabeto Português. Um exemplo de pangrama é: “UM PEQUENO JABUTI XERETA CHAMADO KYA VIU DEZ CEGONHAS FELIZES E GRITOU IWUP, IWUP!”

Construa uma aplicação que recebe uma frase e verifica se ela é pangrama (utilize os conceitos de listas e strings da linguagem C++).

Frases para teste:

- jackdawf loves my big quartz sphinx
- abcdefghijklmnopqrstuvwxyz
- the quick brown fox jumps over a lazy dog
- jackdaws loves my big sphinx of quartz
- hello world
- esta frase es muy larga y contiene todas las letras abc def ghij klmnopqr stu vw x y zzzzz
- supercalifragilistico espialidoso
- alfa beta gamma delta epsilon iotta kappa lambda

```
#include<iostream>
#include<iomanip>
#include<string>
#include<sstream>
#include<locale>
#include<stdlib.h>
#include<stdio.h>
#include<windows.h>
#include"pilha_ligada.h"

using namespace std;

int main()
{
	setlocale(LC_ALL, "Portuguese");

	HANDLE color = GetStdHandle(STD_OUTPUT_HANDLE);

	Pilha_Lig<char> p;
	string frase;
	char c;

	do
	{
		system("cls");

		SetConsoleTextAttribute(color, 15);

		cout << "\n\n\t\t VERIFICADOR DE PANGRAMA" << endl;

		cout << "\n\n\t\t INSIRA A FRASE A SER VERIFICADA: ";
		getline(cin, frase);

		unsigned i = 0;

		int M = 65; // A
		int m = 97; // a

		for (i; i < frase.length(); i++)
		{
			c = frase.at(i);

			if (M != 91) // [ --> primeiro símbolo depois do Z
			{
				if (c == M || c == m)
```

```
                        {
                                p.empilha(c);
                                M++;
                                m++;
                                i = -1;
                        }
                }
                else
                {
                        i = frase.length();
                }
        }
        if (p.elementodotopo() == 'z' || p.elementodotopo() == 'Z')
        {
                SetConsoleTextAttribute(color, 11);
                cout << "\n\n\t\t A FRASE É UM PANGRAMA!!!" << endl;
                Sleep(3000);
        }
        else
        {
                SetConsoleTextAttribute(color, 12);
                cout << "\n\n\t\t A FRASE NÃO É UM PANGRAMA!!!" << endl;
                Sleep(3000);
        }
        SetConsoleTextAttribute(color, 15);
        cout << "\n\n\t\t DESEJA VERIFICAR OUTRA FRASE? S/N ";
        cin >> c;
        cin.ignore();
        if (c != 's' && c != 'S')
        {
                system("cls");
                SetConsoleTextAttribute(color, 15);
                cout << "\n\n\n\n\n\t\t\t FINALIZANDO..." << endl;
                Sleep(2000);
                system("cls");
                SetConsoleTextAttribute(color, 10);
                cout << "\n\n\n\n\n\t\t\t>>>>> PROGRAMA FINALIZADO <<<<<\n\n" << endl;
                SetConsoleTextAttribute(color, 15);
        }
        else
        {
                while (!p.pilhavazia())
                {
                        p.desempilha();
                }
        }
} while (c == 's' || c == 'S');
return 0;
}
```

Header: fila.h

```
#ifndef fila
#define fila

template<typename Tipo>

class Fila
{
private:
	int fim, inic;
	unsigned qnt;
	unsigned tamanho;
	Tipo* v;

public:
	Fila(unsigned tam)
	{
		fim = -1;
		inic = 0;
		qnt = 0;
		tamanho = tam;
		v = new Tipo[tamanho];
	}

	~Fila()
	{
		delete[]v;
	}

	void insere(Tipo x)
	{
		fim++;

		if (fim == tamanho)
		{
			fim = 0;
		}
		v[fim] = x;
		qnt++;
	}

	Tipo remove()
	{
		Tipo rem = v[inic];
		inic++;
		if (inic == tamanho)
		{
			inic = 0;
		}
		qnt--;
```

```
            return rem;
        }

        Tipo primeiro()
        {
            return v[inic];
        }

        bool filacheia()
        {
            return qnt == tamanho;
        }

        bool filavazia()
        {
            return qnt == 0;
        }

        int getFim()
        {
            return fim;
        }

        int getInic()
        {
            return inic;
        }

        int getQnt()
        {
            return qnt;
        }

        Tipo getValor(unsigned pos)
        {
            return v[pos];
        }

        unsigned getTamanho()
        {
            return tamanho;
        }
};

#endif // fila
```

Header: pilha.h

```
#ifndef pilha
#define pilha

template<typename Tipo>

class Pilha
{
private:
	Tipo* v;
	unsigned tamanho;
	int topo;

public:
	Pilha(unsigned tam)
	{
		tamanho = tam;
		v = new Tipo[tamanho];
		topo = -1;
	}

	~Pilha()
	{
		delete[]v;
	}

	void empilha(Tipo x)
	{
		topo++;
		v[topo] = x;
	}

	Tipo desempilha()
	{
		Tipo temp = v[topo];
		topo--;

		return temp;
	}

	Tipo elementodotopo()
	{
		return v[topo];
	}

	bool pilhacheia()
	{
		return topo == tamanho - 1;
	}
```

```
        bool pilhavazia()
        {
                return topo == -1;
        }

        unsigned getTopo()
        {
                return topo;
        }

        unsigned getTamanho()
        {
                return tamanho;
        }

        Tipo getValor(unsigned pos)
        {
                return v[pos];
        }
};

#endif // pilha
```

Header: pilha_ligada.h

```
#ifndef pilha_ligada
#define pilha_ligada

template <typename Tipo>

struct Node
{
	Tipo info;
	Node<Tipo>* prox;
};

template <typename Tipo>

class Pilha_Lig
{
private:
	Node<Tipo>* topo;
public:
	Pilha_Lig()
	{
		topo = NULL;
	}

	bool empilha(Tipo x)
	{
		bool v = true;

		Node<Tipo>* aux = new Node<Tipo>;

		if (aux == NULL)
		{
			v = false;
		}
		else
		{
			aux->info = x;
			aux->prox = topo;
			topo = aux;
		}

		return v;
	}

	Tipo desempilha()
	{
		Tipo temp = topo->info;

		Node<Tipo>* aux = topo;
		topo = topo->prox;
```

```
            delete aux;

            return temp;
        }

        Tipo elementodotopo()
        {
            return topo->info;
        }

        bool pilhavazia()
        {
            return topo == NULL;
        }

        Node<Tipo>* getTopo()
        {
            return topo;
        }

};

#endif // pilha_ligada
```