1. Lista Encadeada com Array

Conceito: Uma lista é uma estrutura de dados que organiza elementos de forma ordenada. Apesar do nome "lista encadeada", aqui ela está implementada sobre um **array**, simulando o comportamento de inserção ordenada, permitindo buscas e remoções.

```
#include <cstdlib>
#include <iostream>

using namespace std;

class ListaDeArray
{
    private:
        int *VET;
        int ProximaPosicaoLivre;
    public:
        ListaDeArray(int qtde)
        {
           VET = new int[qtde];
           ProximaPosicaoLivre = 0;
        }
        void Insere(int n)
        {
            int pos = ProximaPosicaoLivre - 1;
            while((n < VET[pos]) && (pos > -1))
            {
                VET[pos+1] = VET[pos];
                pos--;
            }
            VET[pos+1] = n;
            ProximaPosicaoLivre++;
        }
        void Mostra()
        {
            for(int i = 0; i<ProximaPosicaoLivre; i++)
            {
                cout <<VET[i] <<"\n";
            }
        }
        int Busca(int n)
        {
            for(int i = 0; i<ProximaPosicaoLivre; i++)
            {
                if(n == VET[i]) return i;
```

```
            }
            return -1;
        }
        void Remove(int n)
        {
            int pos = Busca(n);
            if(pos > -1)
            {
                for(int i = pos+1; i < ProximaPosicaoLivre; i++)
                {
                    VET[i-1] = VET[i];
                }
                ProximaPosicaoLivre--;
            }
        }
};

int main(int argc, char *argv[])
{
   ListaDeArray lista(50);
   lista.Insere(1);
   lista.Insere(12);
   lista.Insere(3);
   lista.Insere(7);
   lista.Remove(200);
   lista.Remove(3);
   lista.Mostra();
   cout <<"\n\n\n";
   cout <<lista.Busca(5) <<"\n\n\n";
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

2. Lista com Array

Conceito: Uma lista sequencial baseada em **vetores (arrays)**, que permite inserções e remoções tanto no início quanto no fim. É uma forma simples de manipular coleções de dados.

```
#include <cstdlib>
#include <iostream>

using namespace std;
```

```
class ListaDeArray
{
    private:
        int *VET;
        int ProximaPosicaoLivre;
        int MAX;
    public:
        ListaDeArray(int qtde)
        {
           MAX = qtde;
           VET = new int[MAX];
           ProximaPosicaoLivre = 0;
        }
        void InsereFim(int n)
        {
            if(ProximaPosicaoLivre < MAX)
            VET[ProximaPosicaoLivre++] = n;
        }
        void InsereInicio(int n){

          if(ProximaPosicaoLivre < MAX){

             for(int i=ProximaPosicaoLivre; i>=0; i--){
                   VET[i]=VET[i-1];
              }
              VET[0] = n;
              ProximaPosicaoLivre++;
           }
        }

        void Mostra()
        {
            for(int i = 0; i<ProximaPosicaoLivre; i++)
            {
                  cout <<VET[i] <<"\n";
            }
        }
        int RetiraInicio()
        {
            if(ProximaPosicaoLivre > 0)
            {
               int ValorDeRetorno = VET[0];
               for(int i = 0; i < ProximaPosicaoLivre; i++)
               {
                     VET[i] = VET[i+1];
               }
               ProximaPosicaoLivre--;
```

```
                return ValorDeRetorno;
            }
        }
        int RetiraFim()
        {
            if(ProximaPosicaoLivre > 0)
            return VET[--ProximaPosicaoLivre];
        }
};

int main(int argc, char *argv[])
{
   ListaDeArray lista(50);
   lista.InsereInicio(3);
   lista.InsereInicio(5);
   lista.Mostra();
   cout <<"\nRetira da fila o " <<lista.RetiraFim() <<".\n\n";
   lista.Mostra();
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

3. Fila com Array

Conceito: A fila é uma estrutura de dados do tipo **FIFO (First In, First Out)**, onde o primeiro elemento inserido é o primeiro a ser removido. Muito usada em sistemas de espera e processamento de tarefas.

```
#include <cstdlib>
#include <iostream>

using namespace std;

class FilaDeArray
{
    private:
        int *VET;
        int ProximaPosicaoLivre;
        int MAX;
    public:
        FilaDeArray(int qtde)
        {
            MAX = qtde;
```

```
            VET = new int[MAX];
            ProximaPosicaoLivre = 0;
        }
        void Insere(int n)
        {
            if(ProximaPosicaoLivre <= MAX) VET[ProximaPosicaoLivre++] = n;
        }
        void Mostra()
        {
            for(int i = 0; i<ProximaPosicaoLivre; i++)
            {
                cout <<VET[i] <<"\n";
            }
        }
        int Retira()
        {
            if(ProximaPosicaoLivre > 0)
            {
                int ValorDeRetorno = VET[0];
                for(int i = 1; i < ProximaPosicaoLivre; i++)
                {
                    VET[i-1] = VET[i];
                }
                ProximaPosicaoLivre--;
                return ValorDeRetorno;
            }
        }
};

int main(int argc, char *argv[])
{
    FilaDeArray fila(50);
    fila.Insere(3);
    fila.Insere(5);
    fila.Insere(7);
    fila.Insere(1);
    fila.Mostra();
    cout <<"\nRetira da fila o " <<fila.Retira() <<".\n\n";
    cout <<"\nRetira da fila o " <<fila.Retira() <<".\n\n";
    fila.Mostra();
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

4. Pilha com Array

Conceito: A pilha é uma estrutura de dados do tipo **LIFO (Last In, First Out)**, onde o último elemento inserido é o primeiro a ser removido. É bastante utilizada em algoritmos de recursão e controle de execução.

```
#include <cstdlib>

#include <iostream>


using namespace std;


class PilhaDeArray

{

    private:

        int *VET;

        int ProximaPosicaoLivre;

        int MAX;

    public:

        PilhaDeArray(int qtde)

        {

            MAX = qtde;

            VET = new int[MAX];

            ProximaPosicaoLivre = 0;

        }

        void Empilha(int n)

        {

            if(ProximaPosicaoLivre < MAX)

          VET[ProximaPosicaoLivre++] = n;
```

```
        }
        void Mostra()
        {
            for(int i = 0; i<ProximaPosicaoLivre; i++)
            {
                cout <<VET[i] <<"\n";
            }
        }
        int Desempilha()
        {
            if(ProximaPosicaoLivre > 0)
        return VET[--ProximaPosicaoLivre];
        }
};

int main(int argc, char *argv[])
{
    PilhaDeArray pilha(50);
    pilha.Empilha(3);
    pilha.Empilha(5);
    pilha.Empilha(7);
    pilha.Empilha(1);
    pilha.Mostra();
    cout <<"\nDesempilha " <<pilha.Desempilha() <<".\n\n";
    cout <<"\nDesempilha " <<pilha.Desempilha() <<".\n\n";
    pilha.Mostra();
```

```
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

5. Calendário

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

class Calendario {
private:
    int ano;
    int mes;

public:
    // Construtor
    Calendario(int a, int m) {
        ano = a;
        mes = m;
    }

    bool Bissexto() {
        if (((ano % 4 == 0) && (ano % 100 != 0)) || (ano % 400 == 0))
            return true;
        else
            return false;
    }

    int DiaDaSemana(int dia) {
        int anoAux = ano;
        int f = anoAux + dia + 3 * (mes - 1) - 1;

        if (mes < 3)
            anoAux--;
        else
            f -= int(0.4 * mes + 2.3);

        f += int(anoAux / 4) - int((anoAux / 100 + 1) * 0.75);
        f %= 7;

        return f + 1;
    }

    void ImprimeCalendario() {
        cout << "\n==============================\n";
        cout << "  Calendário de " << mes << "/" << ano << endl;
        cout << "==============================\n\n";
        cout << "DOM\tSEG\tTER\tQUA\tQUI\tSEX\tSAB\n\n";

        short TamanhoDoMes[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

```
if (Bissexto())
    TamanhoDoMes[1] = 29;

for (int j = 1; j < DiaDaSemana(1); j++)
    cout << "\t";

for (int dia = 1; dia <= TamanhoDoMes[mes -
```