Cofinanciado por:

PROGRAMAÇÃO E SISTEMAS DE INFORMAÇÃO
10º ANO

MÓDULO 6 – ESTRUTURAS DE DADOS DINÂMICAS

Escola Profissional BENTO DE JESUS CARAÇA DELEGAÇÃO DO PORTO

Gestão e Programação de Sistemas Informáticos

MÓDULO 6
Estruturas de Dados Dinâmicas

C

1. Introdução
   - 1.1 Conceitos de estruturas Dinâmicas
   - 1.2 Regras de Declaração de Estruturas Dinâmicas

2. Técnicas de manipulação de informação em estruturas dinâmicas

3. Noções de pilha e fila de espera;

4. Operações básicas sobre listas unidirecionais e bidirecionais

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- Compreender o conceito de apontador
- Conhecer as regras de declaração de apontadores
- Identificar as operações para manipulação de apontadores
- Utilizar estruturas dinâmicas lineares
- Distinguir apontador de estrutura dinâmica
- Identificar os tipos de estrutura dinâmica – Pilha e Fila de Espera
- Adquirir a noção de lista bidirecional
- Dominar as operações básicas sobre listas

Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA

MÓDULO 6
Estruturas de Dados Dinâmicas
Programação e Sistemas de Informação 10ºano
C

APONTADORES EM C

- Em C um apontador ou ponteiro (*pointer*) é um tipo especial de variável que é capaz de receber o endereço de memória de outra variável (ficando a apontar para ela).
- O conhecimento do endereço de uma variável pode ser de grande ajuda em certos tipos de rotinas.
- Os apontadores têm duas funções principais em C:

  - Fornecer um meio rápido de **referência aos elementos de um vetor**;
  - Permitir que as funções C **modifiquem os seus parâmetros de chamada.**

MÓDULO 6
Estruturas de Dados Dinâmicas

C

DECLARAÇÃO DE PONTEIROS

- Um ponteiro (como variável que é) tem que ser declarado antes de ser utilizado.
- Formato genérico de declaração: ***Tipo_dados * Nome_variável;***
- EXEMPLO: *int *p; OU int * p; OU int* p;*

Declara a variável ***p*** como um ponteiro para variáveis do tipo *int*.
A variável ***p*** está preparada para receber endereços de variáveis inteiras.

- O * (asterisco) colocado antes do nome da variável indica que esta é um apontador.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO:

- Para declarar uma variável de nome ***p*** do tipo apontador para ***char*** é necessário escrever:

```
char *p;
```

Declara a variável ***p*** como um ponteiro para variáveis do tipo *char*.
A variável ***p*** está preparada para receber endereços de variáveis do tipo carater.

- Para declarar uma variável ponteiro ***x*** para ***floats*** (reais) é necessário escrever:

```
float *x;
```

MÓDULO 6
Estruturas de Dados Dinâmicas

C

OS OPERADORES & E *

- Há dois operadores especiais que permitem a existência dos ponteiros: * e &.

O Operador & (endereço de)

- Operador unário que **devolve o endereço de memória do seu operando**.

EXEMPLO: `p = &cont;`

Coloca em ***p*** o endereço de memória da variável ***cont***. Esse endereço é o posicionamento interno da variável no computador. Nada tem a ver com o valor de ***cont***.

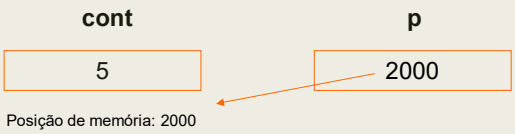
MÓDULO 6
Estruturas de Dados Dinâmicas

Programação e Sistemas de Informação 10ºano

C

Se presumirmos que a variável ***cont*** utiliza a *posição de memória*

2000 para armazenar o seu valor e tiver o *valor 5,* após a atribuição **p=&cont, p** terá o *valor 2000.*



MÓDULO 6
Estruturas de Dados Dinâmicas

C

O Operador * (conteúdo de)

- Operador unário que **devolve o** valor da variável localizada no endereço que se segue.
- EXEMPLO: x =*p;
- Coloca o conteúdo do endereço apontado por ***p*** em ***x*** (valor de ***cont***).
- ***x*** terá o *valor 5* porque 5 está armazenado na *posição 2000* que

é o endereço de memória armazenado em p.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- **EXEMPLO:** *int cont, x, *p;*

  p = &cont;

  *x =*p;*

cont	***p***	***x***
5	2000	5
Posição de memória: 2000	Posição de memória: 2002	Posição de memória: 2004

- **NOTA:** O **operador** and **bit-a-bit** (&) e o **operador multiplicação** (*) têm o mesmo símbolo que estes operadores: * e & mas não há nenhum relacionamento entre si. Tanto & como * têm precedência maior que todos os outros operadores aritméticos, com excepção do menos unário que tem precedência igual.

MÓDULO 6
Estruturas de Dados Dinâmicas

EXEMPLO: o programa seguinte utiliza os operadores **&** e ***** para colocar o *valor 10* numa variável denominada alvo.

```
main0
{
    int fonte, alvo;
    int *p;

    fonte=10;
    p=&fonte;
    alvo=*p;
}
```

- Declara duas variáveis inteiras simples.
- Declara um apontador para uma variável inteira.
- Atribui o valor 10 à variável *fonte*.
- Atribui o endereço da variável fonte ao ponteiro *p*.
- Atribui o conteúdo do endereço contido em *p* à variável alvo.

MÓDULO 6
Estruturas de Dados Dinâmicas

EXEMPLO:

```
int x;
int *p;
```

x

Posição de memória:12FED4

p

Posição de memória: 12FED8

```
x=4;
```

x

4

Posição de memória:12FED4

```
p=&x;
```

p

12FED4

Posição de memória: 12FED8

MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO:

```
#include<iostream>
using namespace std;
main()
{
        int  x, * p;
        x = 4; p = &x;
        cout << &x;
        cout << &p;
        cout << p;
        cout << *p;
}
```

Escreve o endereço de x.

Escreve o endereço de p.

Escreve o endereço de x.

Escreve o valor de x, que é 4.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

ARITMÉTICA DE PONTEIROS

- Existem apenas dois operadores aritméticos que podem ser utilizados nos ponteiros: **+** e – (Soma e Subtracção).

- Cada vez que um ponteiro é incrementado apontara para o próximo local de memória que contem um elemento do seu tipo.

- Cada vez que um ponteiro é decrementado apontará para o local anterior com o elemento daquele tipo.

EXEMPLO:

- Supondo *p1* um **ponteiro para inteiros** com um valor atual de *2000*, após a expressão *p1* ++; o conteúdo de *p1* será *2002* e não *2001*. Para cada incremento de *p1* ele apontará para o inteiro seguinte, que na maioria dos computadores tem dois bytes de comprimento. O mesmo é válido para decréscimos.

- Para ponteiros de carateres isso frequentemente se parece com a aritmética normal.
- Entretanto, todos os outros ponteiros aumentam ou diminuem de acordo com o tipo de dados que apontam.

EXEMPLO:

```
P1 --;
```

Fará com que ***p1*** tenha o valor *1998*, presumindo que o valor anterior era *2000*.

- Não se está limitado a incrementar e decrementar ponteiros. É possível adicionar ou subtrair inteiros aos ponteiros.

- A expressão `p1 = p1+9;` fará com que **p1** aponte para o nono elemento do seu tipo, além daquele que estiver a apontar naquele momento.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- É normal, com a utilização de apontadores, surgirem expressões do tipo:

  *(p+i)

  → Nesta expressão, ao endereço contido por ***p*** são adicionadas ***i*** unidades de endereço.

EXEMPLO:

```
int *p;
*(p+1)
```

→ Nesta expressão, ao endereço contido por ***p*** é adicionada ***1*** unidade de endereço, ou seja, aponta para o endereço seguinte ao endereço apontado por ***p***, mas tendo em conta o tipo de dados.

- Os diferentes tipos de dados podem ocupar diferentes quantidades de memória em bytes. Por isso, em ***(p+1)***, o ***+1*** pode representar 1 byte, 2 bytes, 4 bytes, etc., consoante o tipo de dados em causa.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

REGRAS DA ARITMÉTICA DOS APONTADORES

- Não podem ser executadas outras operações aritméticas sobre ponteiros alem da subtracção e adição entre um ponteiro e um inteiro;
- Não se pode multiplicar ou dividir ponteiros;
- Não se pode somar ou subtrair dois ponteiros;
- Não se pode somar ou subtrair um valor do tipo *float* ou *double* e ponteiros.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

COMPARAÇÃO DE PONTEIROS

- Dois ponteiros que se referem a tipos de variáveis separadas, não apresentam relacionamento entre si.
- Por exemplo, caso *p1* e *p2* sejam ponteiros que apontem para duas variáveis separadas e nao relacionadas, qualquer comparação entre *p1* e *p2* não tem sentido, porque *p1* e *p2* serão obviamente diferentes.
- No entanto, se *p1* e *p2* apontam para variáveis relacionadas entre si, como os elementos de um vetor, *p1* e *p2* poderão ser comparados.

Programação e Sistemas de Informação 10ºano

MÓDULO 6
Estruturas de Dados Dinâmicas

C

APONTADORES E VETORES

- Existe um relacionamento muito estreito entre ponteiros e vetores.
- O compilador lida com o nome do vetor como se tratasse de um ponteiro.
- Qualquer operação que possa ser efetuada com índices de um vetor pode, em alternativa, utilizar apontadores.
- **O endereço do vetor é o mesmo do seu primeiro elemento.**

  `V` OU `&V[0]`
- Um modo comum de utilizar apontadores é inicializando-os para o início do array.

  `int *p = V;` OU `int *p = &V[0];`

MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO:

```
int X[80];
int *p;
p = X;      /* p = & X[0]  */
```

- **p** foi definido como o endereço do primeiro elemento do vetor X.
- Para ter acesso ao quinto elemento de **X,** poderia escrever:

  X[4] ou *(p+4)

MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO:

```
#include<iostream>
using namespace std;
main()
{
    int  V[3] = {2, 5, 9};
    cout << V << '\n';
    cout << &V[0];
    cout << &V[1];
    cout << &V[2];
    cout << V[0];
    cout << *V;
```

Declara V como um vetor de inteiros com 3 elementos e atribui-lhe os valores 2, 5 e 9.

Escreve o endereço do primeiro elemento do vetor V

Escreve o endereço do primeiro elemento do vetor V

Escreve o endereço do segundo elemento do vetor V

Escreve o endereço do terceiro elemento do vetor V

Escreve o valor do primeiro elemento do vetor V = 2

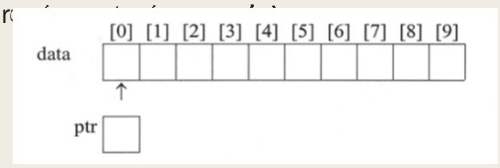
Escreve o conteúdo do endereço do primeiro elemento que é o valor do primeiro elemento do vetor V = 2

MÓDULO 6
Estruturas de Dados Dinâmicas

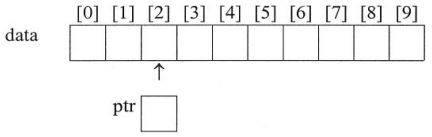
C

EXEMPLO: O fragmento de código seguinte mostra que o ponteiro **ptr** aponta para o primeir

```
char data[10], *ptr;
ptr = data;
```



A expressão: **ptr+=2;** Aponta para o terceiro elemento do array *data*.

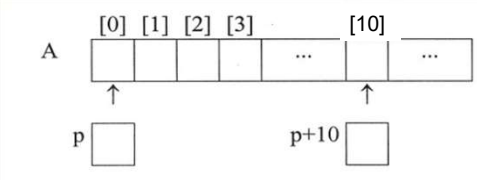


MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO: O fragmento de código seguinte mostra que o ponteiro ***p*** aponta para o primeiro elemento do array ***a*** e que é possível manipular os elementos de um vetor utilizando a aritmética de apontadores.

```
int a[64], *p;
p=&a[0];     OU    p=a;
```



A expressão p+10 é um apontador de a[10].

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- EXEMPLO: Programa de vetores implementado com índices.

```
#include<iostream>
using namespace std;
main()
{
        int  i, v[3] = {2, 5, 9};
        for (i=0 ; i<3 ; i++)
        {
              cout << v[i] << '\n';
        }
}
```

Declara um vetor ***v*** com 3 elementos e atribui os valores 2, 5 e 9 ao vetor.

Imprime os elementos do vetor em linhas diferentes.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- EXEMPLO: Programa de vetores implementado com apontadores.

```
#include<iostream>
using namespace std;
main()
{
        int  i, v[3] = {2, 5, 9};
        for (i=0 ; i<3 ; i++)
        {
              cout << *(v + i) << '\n';
        }
}
```

Declara um vetor ***v*** com 3 elementos e atribui os valores 2, 5 e 9 ao vetor.

Imprime os elementos do vetor em linhas diferentes, considerando ***v*** (nome do vetor) como um ponteiro constante que aponta para o primeiro elemento do vetor.

(v+i) é equivalente ao endereço de v[i]

****(v+i)*** é equivalente ao valor de v[i]

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- EXEMPLO: Programa de vetores implementado com apontadores.

```
#include<iostream>
using namespace std;
main()
{
      int  i, *p, v[3] = {2, 5, 9};
      p = v;
      for (i=0 ; i<3 ; i++)
      {
           cout << *(p + i) << '\n';
      }
}
```

Declara um vetor ***v*** com 3 elementos e atribui os valores 2, 5 e 9 ao vetor.

Imprime os elementos do vetor em linhas diferentes, utilizando um ponteiro *p*, ao qual é atribuído o endereço do vetor.

p = v; → ***p*** fica com o endereço do vetor

(p+i) é equivalente ao endereço de v[i]

****(p+i)*** é equivalente ao valor de v[i]

Programação e Sistemas de Informação 10ºano

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- Em resumo, C permite dois métodos de acesso aos elementos de um vetor:

  - indexação

  - ponteiros

- A aritmética dos ponteiros pode ser mais rápida que a indexação do vetor.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

APONTADORES E STRINGS

- Uma vez que uma string em C é um vetor de carateres, é possível manipular uma string utilizando apontadores.

- EXEMPLO:

```
char s[ ] = "Hello";
```
→ A Variável é declarada explicitamente como um array de *char*.

OU

```
char *s = "Hello";
```
→ A Variável é declarada como um apontador para *char*.

- Em ambas as situações a variável é declarada com a sua inicialização.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- Uma **vantagem** da utilização de apontadores na declaração de strings é que estas podem ser declaradas sem inicialização e receber strings com diferentes tamanhos.

- EXEMPLO:

```
char *s;
s = "Hello";
cout << s << '\n';
s = "Hello World";
cout << s << '\n';
```

MÓDULO 6
Estruturas de Dados Dinâmicas

C

char s[ ] = 'Hello'

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]
'H'	'e'	'l'	'l'	'o'	'\0'

char *s = 'Hello'

*s	*(s+1)	*(s+2)	*(s+3)	*(s+4)	*(s+5)
'H'	'e'	'l'	'l'	'o'	'\0'

s

MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO: Estes programas imprimem a string ***s*** carater a carater.

```
#include <iostream>
using namespace std;
main()
{
      char s[ ] = "Hello";
      for ( int i = 0; s[i] != '\0'; i++)
      {
           cout << s[i] << '\n';
      }
}
```

```
#include <iostream>
using namespace std;
main()
{
     char *s = "Hello";
     for ( int i = 0; *(s+i) != '\0'; i++)
     {
       cout << *(s+i) << '\n';
     }
}
```

MÓDULO 6
Estruturas de Dados Dinâmicas

C

```
#include <iostream>
using namespace std;
main()
{
    char *s = "Hello";
    while (*s != '\0')
    {
      cout << *s++ << '\n';
    }
}
```

```
#include <iostream>
using namespace std;
main()
{
    char *s;
    s = "Hello";
    cout << s << '\n';
    s = "Hello World";
    cout << s << '\n';
    while (*s) cout << *s++ << '\n;
}
```

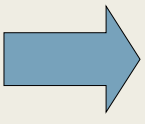

MÓDULO 6
Estruturas de Dados Dinâmicas

C

APONTADORES PARA ESTRUTURAS

- O endereço de uma variável estrutura é do tipo **apontador para estrutura**.
- Se a variável ***p*** for declarada como:

```
1. struct date
     int year;
     int month;
     int day;
} *p;
```



Então as construções:

(*p).year
(*p).month
(*p).day

são comuns em C!

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- Existe um operador especial para o acesso aos elementos da estrutura. É o operador → que é formado utilizando um sinal menos (-) e um sinal maior (>).
- O sinal → é utilizado em lugar do operador ponto **(.)** ao aceder a um elemento de uma estrutura numa função ou num vetor representado por um apontador.
- Assim,

(*p).year	p → year
(*p).month	p → month
(*p).day	p → day

MÓDULO 6
Estruturas de Dados Dinâmicas

C

O operador → é definido como:

apontador → elemento_da_estrutura

O que é equivalente a:

(*apontador). elemento_da_estrutura

MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO:

```
#include <iostream>
using namespace std;
main()
{    struct aluno
     { char nome[20];
       int idade;
     } a1, a2;
     struct aluno *p;
     p = &a1;
     strcpy( (*p).Nome, "Ana Cruz") );
     (*p).idade = 17;
     cout << (*p).nome << '\t' << (*p).idade << '\n';
     p =&a2;
     strcpy( p→Nome, "Eva Silva");
     p →idade =18;
     cout << p → nome << '\t' << p → idade << '\n';

}
```


Programação e Sistemas de Informação 10ºano

MÓDULO 6
Estruturas de Dados Dinâmicas

C

ALOCAÇÃO DINÂMICA

- Há duas formas básicas de atribuir memória RAM para os dados com que um determinado programa trabalha:

1. *Através da declaração de variáveis fixas.*
2. *Através da atribuição dinâmica de memória.*

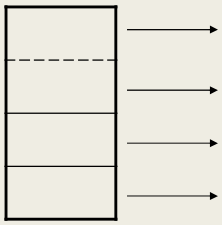
Permite que um programa solicite e consiga novos Blocos de memória para os novos dados.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- Em C é possível alocar e libertar memória dinamicamente utilizando algumas funções da biblioteca padrão.

- Estrutura de um programa executável (*.EXE):



STACK: Onde são armazenados os patâmetros de Input/Otput das funções (valores de retorno)

HEAP: Onde são armazenadas as variáveis locais

Código Executável

ZONA DE DADOS ESTÁTICA: Onde são armazenadas as variáveis globais e as variáveis estáticas locais.

MÓDULO 6
Estruturas de Dados Dinâmicas

Programação e Sistemas de Informação 10ºano

C

Variáveis Globais (e Variáveis estáticas Locais)

- São variáveis que existem durante todo o programa. São declaradas com completa informação da sua estrutura (tipo e tamanho).

- EXEMPLO:

```
int a;
char b[20];
main()
{
        ...
}
```

Estas variáveis existirão durante toda a execução do programa sem qualquer variação no seu tamanho e podendo ser acedidas em qualquer altura.

Estas variáveis têm carácter estático e por isso o compilador reserva espaço fixo de memória para elas. (ZONA DE DADOS ESTÁTICA)

MÓDULO 6
Estruturas de Dados Dinâmicas

C

Variáveis de tamanho indeterminado

- Um programa não pode reservar mais espaço de memória para os dados com que opera do que aquele que foi previsto.
- Por vezes torna-se necessário atribuir mais memória para novos dados que entretanto podem surgir durante a interação com o utilizador.
- È frequente, num programa, o programador não ter informação completa acerca das variáveis que utiliza.

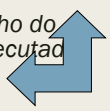
MÓDULO 6
Estruturas de Dados Dinâmicas

C

Por exemplo:

- *Num programa que utilize um vetor, em que o vetor é declarado com um número fixo de elementos.*
- *Quando o vetor é declarado, sendo o seu tamanho desconhecido, ele será sobredimensionado (levando a desperdício de memória) ou subdimensionado (levando à inadequação do programa)*
- *Nestes casos seria desejável determinar o tamanho do vetor apenas quando o programa já está a ser executado*

É possível com a alocação de memória!



MÓDULO 6
Estruturas de Dados Dinâmicas

C

- Em C existem duas funções que permitem a alocação dinâmica de memória:

malloc() Função que obtém um bloco de memória com um tamanho que é necessário indicar na instrução e que é atribuído a um apontador que será convertido para o tipo de dados a trabalhar.

```
SINTAXE:  p = malloc( num * sizeof(tipo_dado))
```

free() Função que liberta a memória obtida com a função anterior.

```
SINTAXE:  free(p)
```


MÓDULO 6
Estruturas de Dados Dinâmicas

C

Operador sizeof()

- Apresenta o tamanho em bytes de qualquer tipo de dados, incluindo estruturas.
- SINTAXE: `sizeof(nome_variável)`

OU

```
sizeof(tipo_dados)
```

MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO:

```
main()
{
    int n;
    struct alunos *v;
    printf("Quantos alunos?"); scanf("%d", &n);
    v = malloc( n * sizeof(struct alunos));
    ...
}
```


MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO:

```
main()
{
    int n;
    int *v;
    printf("Quantos alunos?"); scanf("%d", &n);
    v = malloc( n * sizeof(int));
    ...
}
```

EXEMPLO:

```
main()
{
    int n;
    printf("Quantos alunos?");
    scanf("%d", &n);
    int v[n];
     …
}
```

MÓDULO 6
Estruturas de Dados Dinâmicas

C

- Em C existem dois operadores que permitem a alocação dinâmica de memória:

new Este operador obtém um bloco de memória do sistema com o tamanho necessário para o tipo de dados que foi solicitado e retorna um ponteiro para o início desse bloco de memória.

SINTAXE: var_ponteiro = **new** tipo_dados;

delete Operador contrário ao anterior que apaga/liberta a memória anteriormente obtida com o operador *new*.

SINTAXE: ***delete*** var_ponteiro;

MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO: (variável simples)

```
main()
{
      int *p;
      p = new int;
      cout << "Introduza um número inteiro:"
      cin >> *p;
      cout << "Bytes de um número inteiro:" << sizeof (int)
      delete p;
}
```

MÓDULO 6
Estruturas de Dados Dinâmicas

C

EXEMPLO: (vetor de números inteiros)

```
main()
{
        int *v, i, soma=0;
        cout << "Quantos elementos?" ;
        cin >> n;
        v = new int[n];
        for(i=0; i<n; i++)
        {       cout << "Introduza um número inteiro:";
                cin >> *(v+i);                                  //ou v[i]
                soma = soma + *(v+i)
        }
        cout << "A Soma é:" << Soma
        delete [] v;
}
```

O ponteiro v fica a apontar para um bloco de memória com tamanho n.

MÓDULO 6
Estruturas de Dados Dinâmicas

C

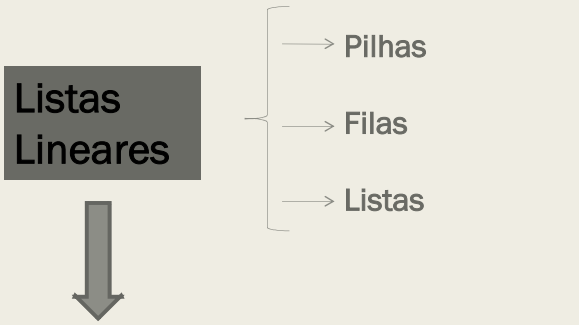
EXEMPLO: (Estruturas)

```
main()
{
        struct pessoa {
                char nome[20];
                int idade;
        }p1, p2;
        strcpy(p1.nome, "Ana Silva");
        p1.idade=16;
        strcpy(p2.nome, "André Santos");
        p2.idade=18;
        cout << p1.nome << '\t';
        cout << p1.idade << '\t';
        cout << p2.nome << '\t';
        cout << p2.idade << '\t';
        ...
        }
```

```
…
pessoa *p;
p = new pessoa;
strcpy( p->nome, "Ema Sá");
p->idade =17;
cout << p->nome << '\t';
cout << p->idade << '\t';
```

MÓDULO 6
Estruturas de dados dinâmicas

C



Estruturas de dados de comprimento variável.

MÓDULO 6
Estruturas de dados dinâmicas

Programação e Sistemas de Informação 10º

C

PILHA

❑ Uma pilha é uma **lista linear** onde as operações de inserção e remoção (todos os acessos) são efetuados apenas numa extremidade da lista, ou seja:

- A inserção de um elemento X torna-o o último elemento da lista;
- A remoção é sempre efetuada sobre o último elemento da lista.

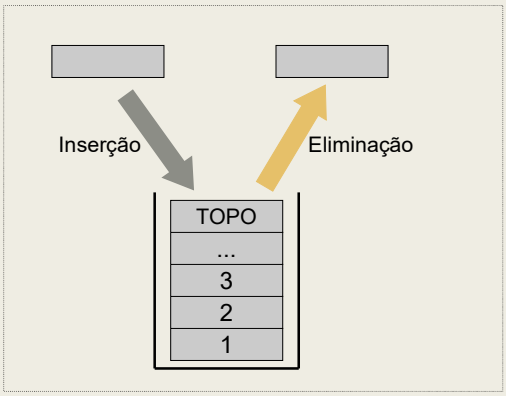
❑ Devido às caraterísticas das operações da pilha, o último elemento a ser inserido será o primeiro a ser retirado.

❑ Estruturas deste tipo são conhecidas como "**LIFO**" (*last in, first out*).

MÓDULO 6
Estruturas de dados dinâmicas

C

PILHA



"LIFO" (*last in, first out*)

O último elemento a entrar é o 1º a sair.

As operações de adição e eliminação só podem ser executadas num dos lados da pilha.

Os elementos só podem ser removidos na ordem inversa aquela em que foram adicionados.

MÓDULO 6
Estruturas de dados dinâmicas

C

FILA

❑ Uma fila é uma **lista linear** onde as operações de inserção são efetuadas num extremo da lista e as operações de eliminação no outro extremo da lista, ou seja:

- A inserção de um elemento torna-o no último elemento da lista;
- A eliminação é sempre efetuada sobre o primeiro elemento da lista.

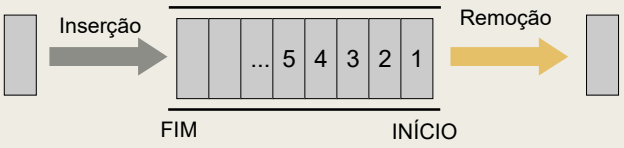
❑ Devido às caraterísticas das operações da fila, **o primeiro elemento a ser inserido será o primeiro a ser retirado**.

❑ Estruturas deste tipo são conhecidas como "**FIFO**" (*first in, first out*).

MÓDULO 6
Estruturas de dados dinâmicas

C

FILA



- Inserções → enfileirar = pela retaguarda
- Eliminações → desenfileirar = pela frente

"FIFO" (*first in, first out*)

O primeiro elemento a entrar é o 1º a sair.

MÓDULO 6
Estruturas de dados dinâmicas

C

LISTA ENCADEADA

❑ Uma lista encadeada é uma coleção ordenada de registos em que cada registo possui um campo que indica a localização do registo seguinte na lista.

❑ O primeiro registo é chamado a cabeça da lista.

❑ Vantagens das listas encadeadas:

- ✓ Permitem usar o espaço de memória necessário sem precisar de reservar espaço para o pior caso;
- ✓ A memória pode ser reservada e libertada conforme for sendo necessário durante o programa. Isto porque as listas ligadas são implementadas usando variáveis dinâmicas;
- ✓ Grande rapidez nas operações de remoção e inserção de registos quando comparado com as listas sequenciais.

MÓDULO 6
Estruturas de dados dinâmicas

C

LISTA ENCADEADA - Tipos

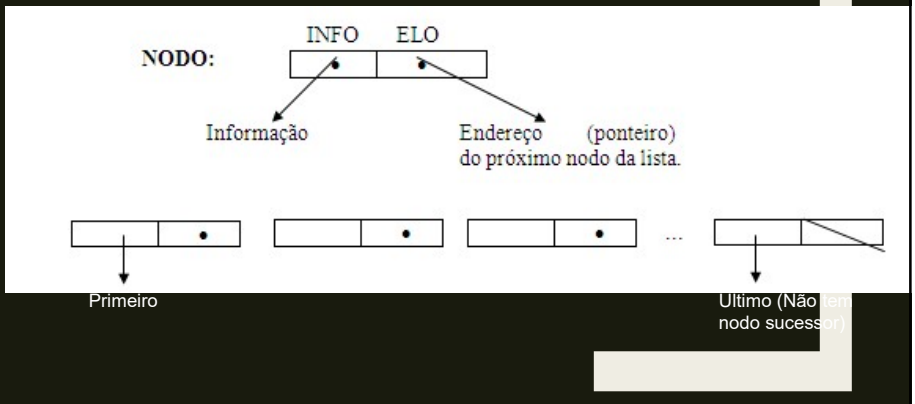
As listas podem ser:

Simplesmente encadeada - lista que contém apenas o endereço de um registo.

Duplamente encadeada - lista que contém os endereços dos registos anterior e posterior.

MÓDULO 6
Estruturas de dados dinâmicas

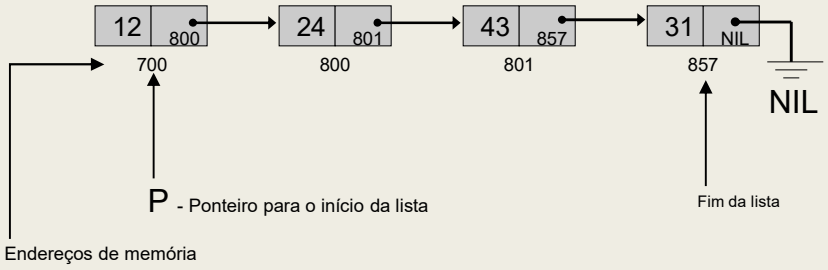
LISTA ENCADEADA



MÓDULO 6
Estruturas de dados dinâmicas

C

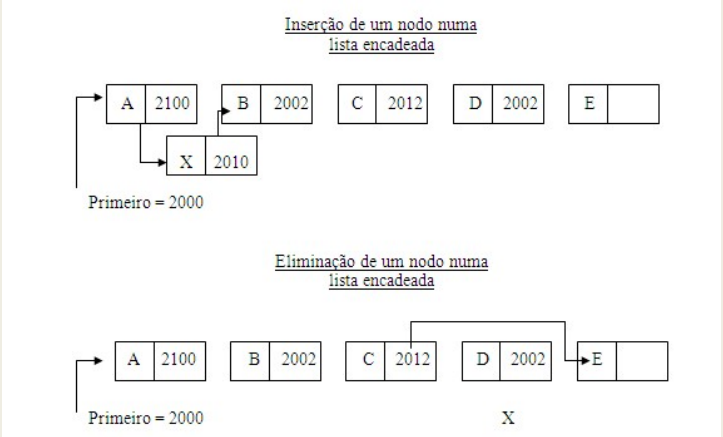
LISTA ENCADEADA



MÓDULO 6
Estruturas de dados dinâmicas

C

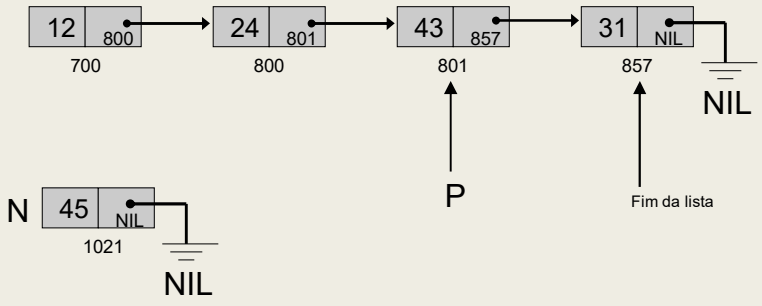
LISTA ENCADEADA



MÓDULO 6
Estruturas de dados dinâmicas

C

- Inserção
  - *Inserção de um novo nó **N** na posição apontada por **P**.*

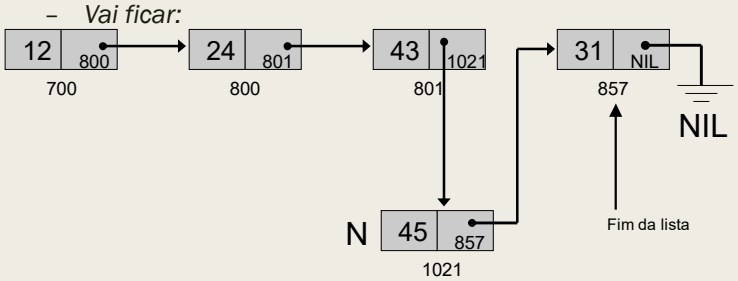


MÓDULO 6
Estruturas de dados dinâmicas

Programação e Sistemas de Informação 10º

C

- Inserção
  - *Vai ficar:*



MÓDULO 6
Estruturas de dados dinâmicas

C

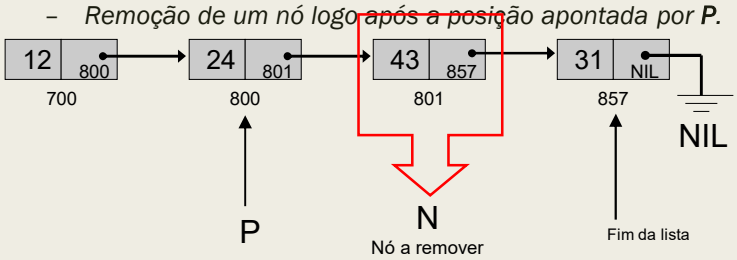
- **Inserção**
  - *Guardar o endereço do próximo nó;*
  - *O nó que está a ser apontado por **P** passa a ser **N**.*

Programação e Sistemas de Informação 10º

MÓDULO 6
Estruturas de dados dinâmicas

C

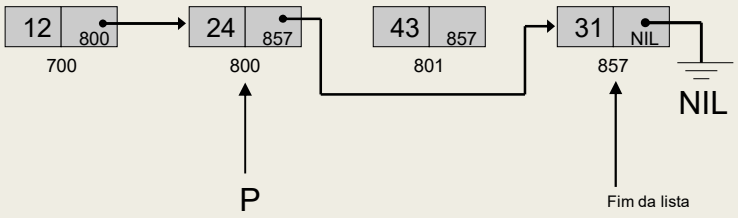
- **Remoção**
  - *Remoção de um nó logo após a posição apontada por **P**.*



MÓDULO 6
Estruturas de dados dinâmicas

C

- Remoção
  - *Vai ficar:*



Programação e Sistemas de Informação 10º

MÓDULO 6
Estruturas de dados dinâmicas

C

- Remoção
  - *Guardar o endereço do nó que será removido num apontador auxiliar **N**;*
  - *Fazer o nó apontar para o que o **N** aponta;*
  - *Libertar **N**.*

MÓDULO 6
Estruturas de dados dinâmicas

C

- **BIBLIOGRAFIA**

  - *AZUL, Artur Augusto (2004), Bases de Programação 10º, Lisboa, Porto Editora*
  - *SIMÕES, Francisco et al (2004), Bases de Programação 10º, Porto, Editora ASA*
  - *TREMBLAY, Jean-Paul, Richard B. BUNT, Ciência dos Computadores uma abordagem algoritmica, Madrid, Editora McGraw-Hill*
  - *GUERREIRO, Pedro, Pascal – Técnicas de Programação*
  - *AZUL, Artur Augusto (2010), LP / PSI, Porto, Porto Editora*