

Estruturas de Dados

Engenharia Informática

1º Ano - 2º Semestre

Francisco Morgado

Escola Superior de Tecnologia e Gestão de Viseu

2012-2013

3. ESTRUTURAS DINÂMICAS

3.1 Pilhas (Stacks)

3.2 Filas de espera (Queues)

3.3 Listas ligadas ordenadas

3.4 Listas bi-ligadas ordenadas

3. ESTRUTURAS DINÂMICAS

3.1 Pilhas (Stacks)

Conceito

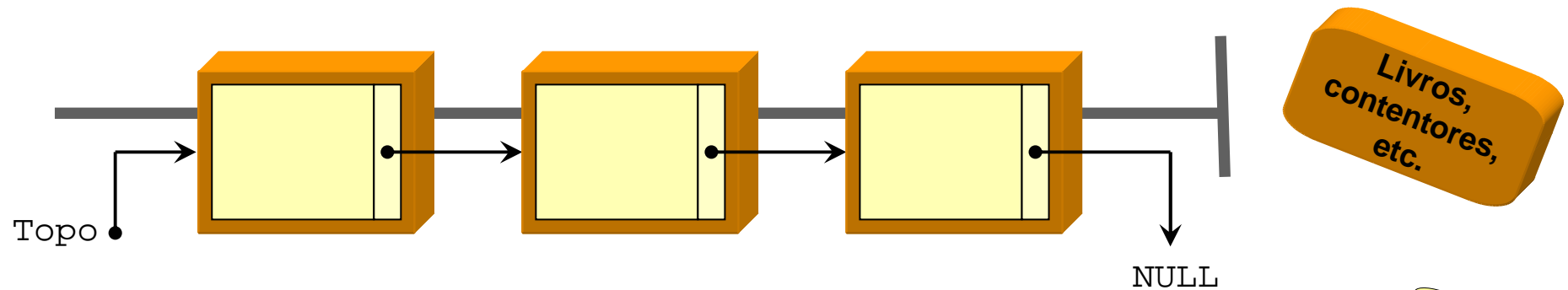
Inserção de elementos

Remoção de elementos

Listagem de todos os elementos

Aplicação

Pilhas (Stacks) - Conceito



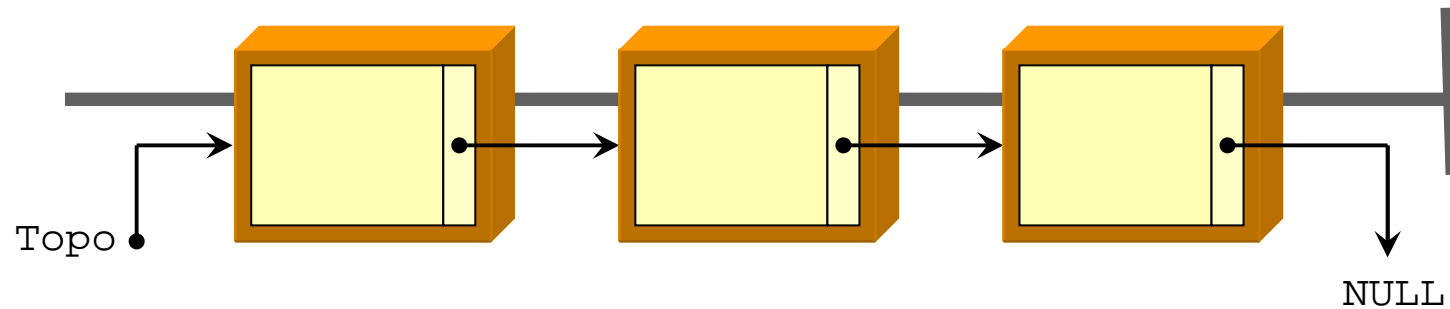
```
typedef struct
{
    char conteudo[80];
    char origem[30];
    char destino[30];
    float tonelagem;
    int referencia; // chave
} Informacao;
```

■ Operações na pilha

- ❑ Inserir elemento no topo
- ❑ Retirar elemento do topo
- ❑ Listar elementos na pilha
- ❑ Contar n.º de elementos na pilha

LIFO

Pilhas (Stacks) - Conceito



```
typedef struct nodo
{
    Informacao dados;
    struct nodo* seguinte;
}Nodo;
```

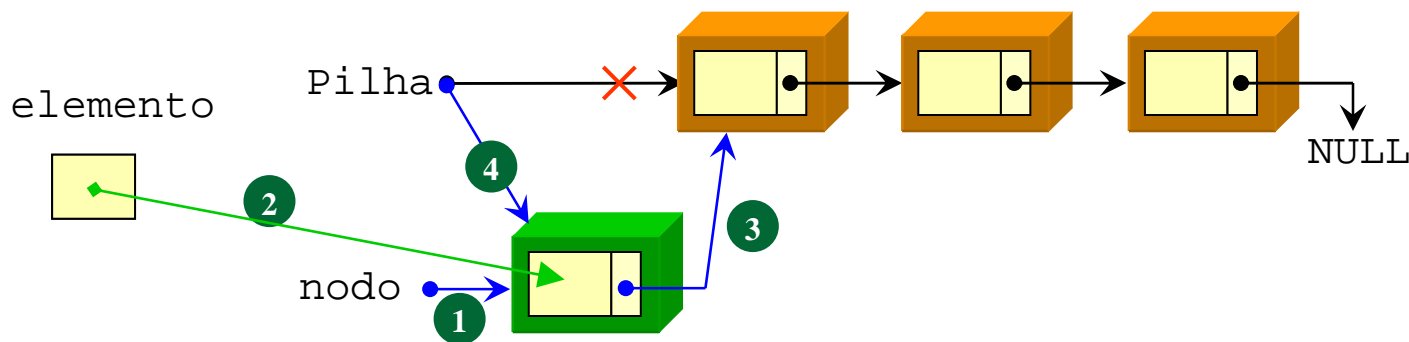
```
typedef Nodo* Pilha;
```

Criar pilha vazia

```
Pilha pilha;
```

Pilhas (Stacks) - Inserção de elementos

```
void inserir(Informacao elemento, Pilha* pilha)
{
    1 Nodo* nodo = (Nodo*) malloc(sizeof(Nodo));
    2 nodo->dados = elemento;
    3 nodo->seguinte = *pilha;
    4 *pilha = nodo;
}
```



Chamada
`inserir(elemento, &topo);`

Pilhas (Stacks) - Remoção de elementos

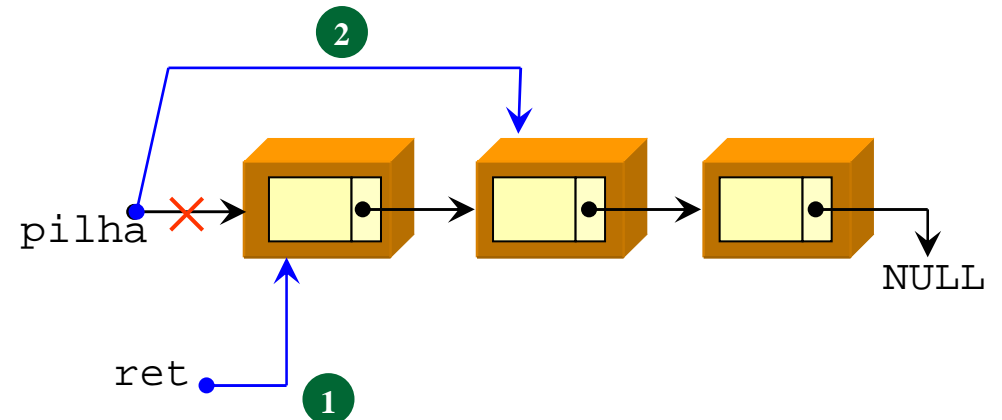
```
Nodo* remover(Pilha* pilha)
{
    Nodo* ret;
    if (*pilha == NULL) //se a pilha
        return NULL;   //está vazia
```

1 ret = *pilha;

2 *pilha = (*pilha)->seguinte;
//topo passa a apontar para o novo 1º
return ret;
// apontador para o nodo retirado
}

Chamada

nodo = remover(&topo);



Pilhas (Stacks) - Listagem de todos os elementos

```
void listar(Pilha pNodo)
{
    int i = 1;
    while (pNodo)
    {
        printf("\nRegistro %d:", i++);
        mostrar(pNodo->dados);
        pNodo = pNodo->seguinte;
    }
}
```

Idioma		
Apontador	int	bool
NULL	0	false
≠ NULL	≠ 0	true

Chamada
listar(topo);

```
void mostrar(Informacao elemento)
{
    printf("\nConteúdo:%s", elemento.conteudo);
    ...
    printf("\nReferência:%d", elemento.referencia);
}
```


Aplicação a operações sobre uma pilha de livros

Estrutura

```
typedef struct livro
{
    char titulo[50];
    char registro[10];
}Livro;
```

```
typedef struct nodo
{
    Livro dados;
    struct nodo* seguinte;
}Nodo;

typedef Nodo* Pilha;
```

```
typedef Livro Informacao;
```

Alocação de um livro

```
Nodo* nodoLivro = (Nodo *)malloc(sizeof(Nodo));
```

Apontador para o registro criado

Leitura de dados relativos a um livro

```
Livro lerDados()  
{  
    limparTeclado();  
    Livro livro;  
    printf("\nQual o Titulo?");  
    gets(livro.titulo);  
    printf("\nQual o Registo?");  
    gets(livro.registo);  
  
    return livro;  
}
```

```
void limparTeclado()  
{  
    fflush(stdin); //Visual Studio  
}
```

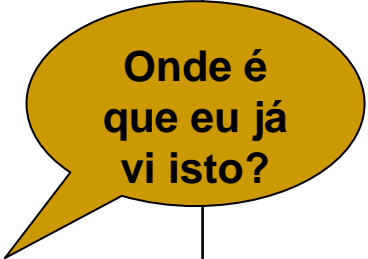
Escrita de dados relativos a um livro

```
void mostrar(Livro livro)
{
    printf("\nTitulo: %s\tRegistro: %s", livro.titulo,
          livro.registo);
}
```

```
void listar(Pilha pNodo)
{
    int i = 0;
    while (pNodo != NULL)
    {
        printf("\n%do Livro ", ++i);
        mostrar(pNodo->dados);
        pNodo = pNodo->seguinte;
    }
}
```

Inserção de um livro (no topo da pilha)

```
void inserir(Livro livro, Pilha* topo)
{
    Nodo* nodo = (Nodo*)malloc(sizeof(Nodo));
    nodo->dados = livro;
    nodo->seguinte = *topo;
    *topo = livro;
}
```



Onde é
que eu já
vi isto?

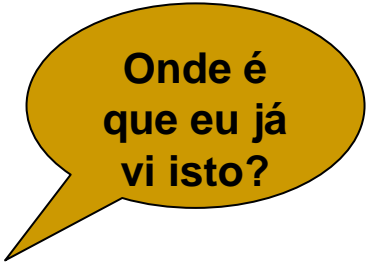
Remoção de um livro (do topo da pilha)

```
Nodo* remover(Pilha* topo)
{
    Nodo* rem;

    if (*topo == NULL)
        return NULL;

    rem = *topo;
    *topo = (*topo)->seguinte; //topo passa a apontar para o novo 1º

    return rem;
}
```



Onde é
que eu já
vi isto?

Determinar a altura da pilha (número de livros)

Passagem por
valor

```
int altura(Pilha topo)
{
    int i = 0;
    while (topo != NULL)
    {
        topo = topo->seguinte;
        i++;
    }
    return i;
}
```

Função main()

```
void main()
{
    int opcao;
    Livro livro;
    Pilha topo = NULL;
    Nodo* nodo;

    do
    {
        opcao = menu();    // EXERCÍCIO: ESCREVER A FUNÇÃO int menu()
        switch (opcao)
        {
            case 1: livro = lerDados();
                    inserir(livro, &topo);
                    break;
            case 2: nodo = remover(&topo);
                    if (nodo)
                    {
                        mostrar(nodo->dados);
                        free(nodo);
                    }
                    break;
            case 3: listar(topo);
                    break;
            case 4: printf("\nAltura da pilha de livros:  %d", altura(topo));
                    break;
            case 0:  exit(EXIT_SUCCESS);      }
        }
    } while (1);
}
```

3.2 Filas de espera (Queues)

Conceito

Inserção de elementos

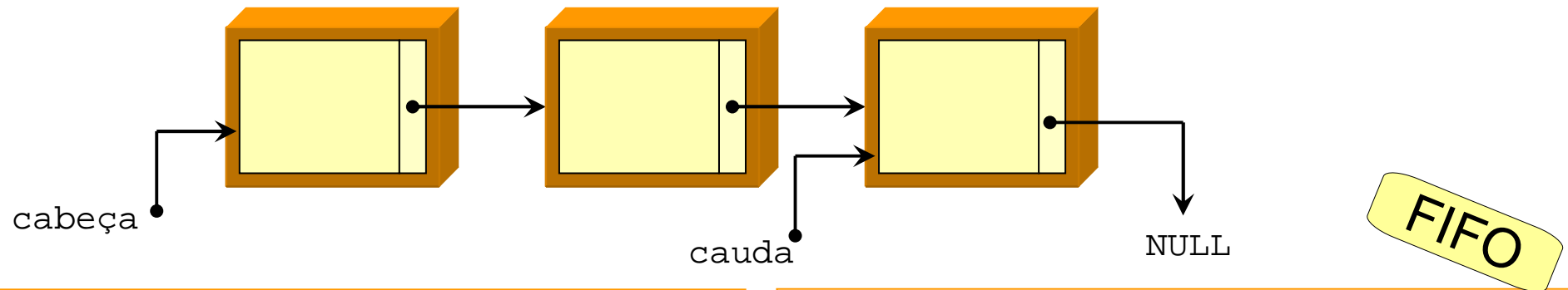
Remoção de elementos

Listagem de todos os elementos

Algumas ferramentas para processamento de tempo

Operações numa fila de espera de alunos

Filas de Espera - Conceito



```
typedef struct
{
    char nome[80];
    char bi[12];
    int nMec; /* chave */
} Informacao;
```

```
typedef struct
{
    Nodo* cabeca;
    Nodo* cauda;
} Fila;
```

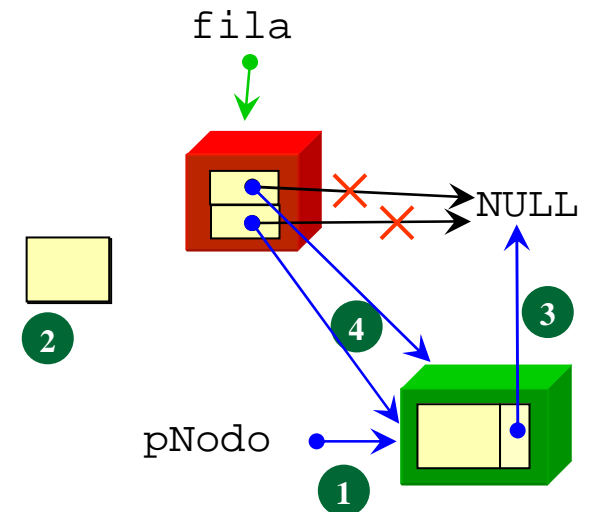
■ Operações na Fila

- ❑ Inserir elemento no fim (cauda)
- ❑ Retirar do início (cabeça)

```
typedef struct nodo
{
    Informacao dados;
    struct nodo* seguinte;
} Nodo;
```

Filas de Espera (Queues) - Inserção de elementos

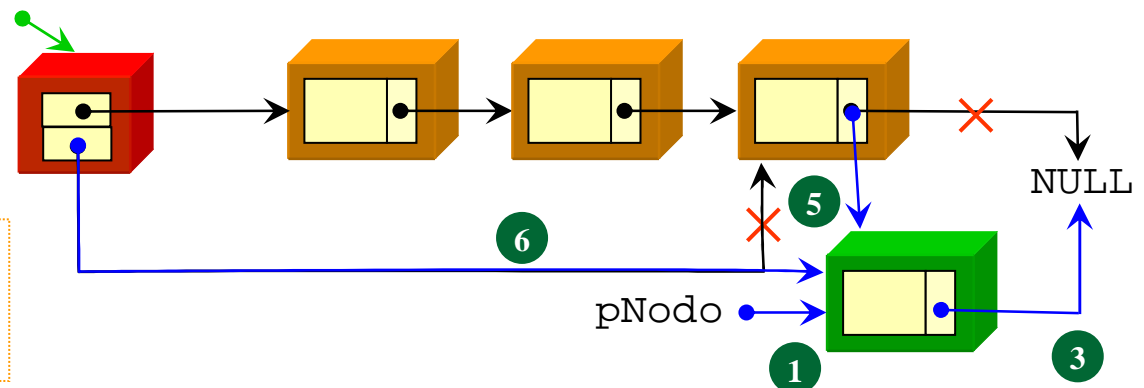
```
void inserir(Informacao elemento, Fila* fila)
{
1  Nodo* pNodo = (Nodo*)malloc(sizeof(Nodo));
2  pNodo->dados = elemento;
3  pNodo->seguinte = NULL;
4  if (fila->cabeca == NULL) //se fila vazia
    fila->cabeca = fila->cauda = pNodo;
5  else
6  { //o ex-último aponta para o novo último
    (fila->cauda)->seguinte = pNodo;
    (fila->cauda) = pNodo; // cauda aponta
    // para o novo último
  }
}
```



elemento fila



2



Chamada

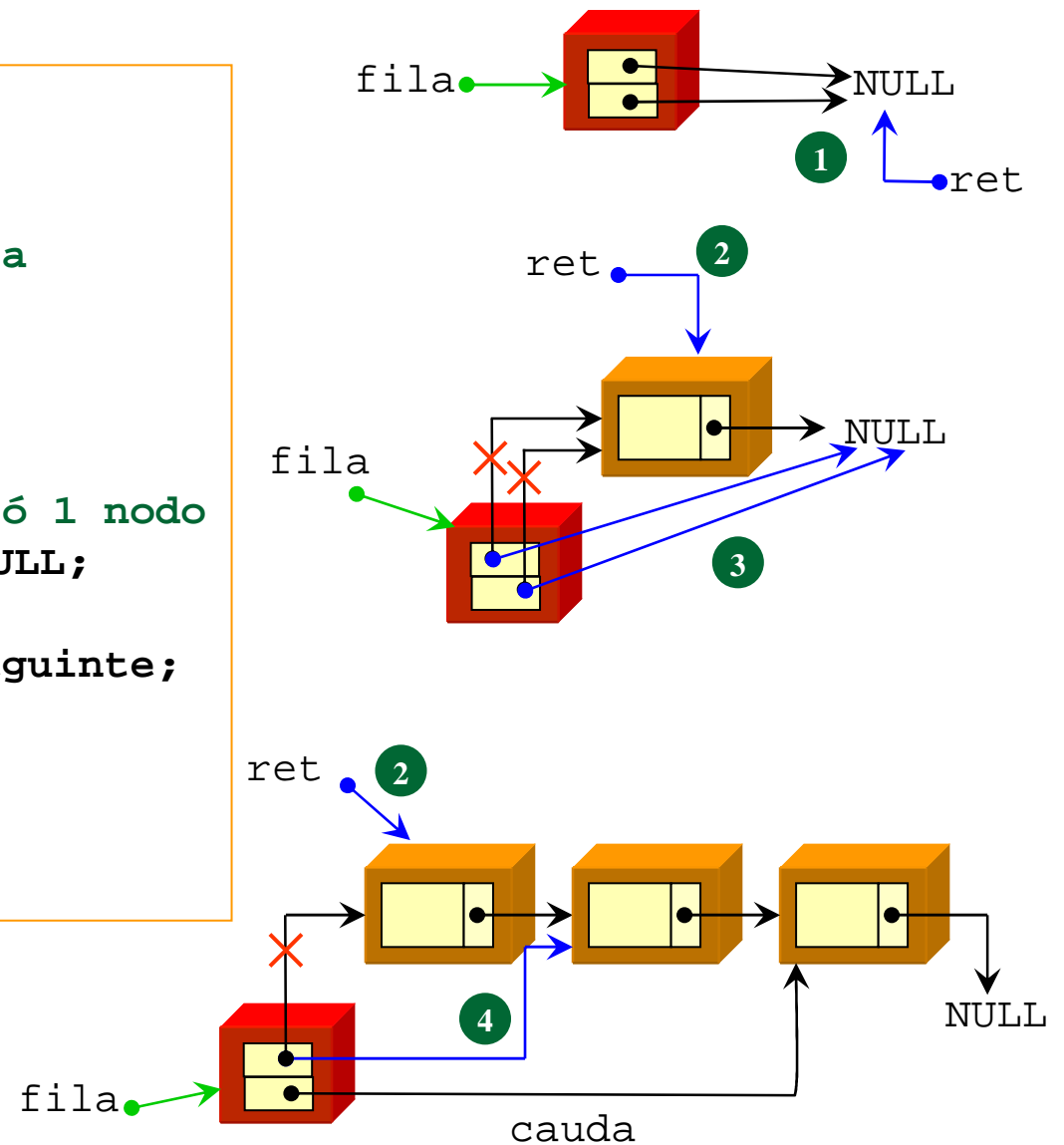
`inserir(elemento, &fila);`

Filas de Espera - Remoção de elementos

```
Nodo* remover(Fila* fila)
{
    Nodo* ret;
    if (fila->cabeca == NULL) //fila vazia
    ① ret = NULL;
    else
    {
        ② ret = fila->cabeca;
        if(fila->cabeca == fila->cauda) //só 1 nodo
        ③     fila->cabeca = fila->cauda = NULL;
        else
        ④     fila->cabeca = (fila->cabeca)->seguinte;
            // cabeca passa a apontar
            // para o novo 1º elemento
    }
    return(ret);
}
```

Chamada

pNodo = remover(&fila);



Filas de Espera – Listagem dos elementos

```
void listar(Fila fila)
{
    int i = 1;
    Nodo* pNode = fila.cabeca;
    while (pNode != NULL)
    {
        printf("\n %dº Elemento:", i++);
        mostrar(pNode->dados);
        pNode = pNode->seguinte;
    }
}
```

Chamada
listar(fila);

```
void mostrar(Informacao elemento)
{
    printf("\n  Nome:%s",      elemento.nome);
    printf("\n  BI:%s",        elemento.bi);
    printf("\n  N°Aluno:%d",   elemento.nMec);
}
```

Algumas ferramentas para processamento de tempo

time_t

- tipo Time (`time_t`) permite a representação de tempos e possibilita a realização de operações aritméticas.
- `time_t` é a contagem do número de segundos desde 1/Jan/1970

struct tm

A estrutura Time (`struct tm`) contém componentes do tempo e do calendário, de tipo `int`:

Data e tempo numa versão mais *human-friendly*!

Campo	Significado	Valores (inclusive)
<code>tm_sec</code>	Segundos após o minuto	0 a 59
<code>tm_min</code>	Minutos após a hora	0 a 59
<code>tm_hour</code>	Horas após a meia-noite	0 a 23
<code>tm_mday</code>	Dia do mês	1 a 31
<code>tm_mon</code>	Meses após Janeiro	0 a 11
<code>tm_year</code>	Anos após 1900	0 a INT_MAX
<code>tm_wday</code>	Dias após Domingo	0 a 6
<code>tm_yday</code>	Dias após 1 de Janeiro	0 a 365

Algumas funções

time_t time(**time_t*** timePointer);

Devolve o tempo do calendário corrente ou -1 se não estiver disponível. Se timePointer não for NULL, o valor devolvido é também atribuído a *timePointer.

```
/* Exemplo de utilização da função time */
#include <stdio.h>
#include <time.h>
void main ()
{
    long segundos = time(NULL);
    printf ("\n%ld segundos desde 1 de Janeiro de 1970", segundos);
    printf ("\n%ld horas desde 1 de Janeiro de 1970", segundos/3600);
}
```

double difftime(**time_t** time2, **time_t** time1);

Devolve a diferença, em segundos, entre time2 e time1.

struct tm* localtime(const **time_t*** timePointer);

Devolve o tempo do calendário *timePointer, convertido em tempo local.

Exemplos de aplicação

```
int tempoEspera(time_t fim, time_t inicio)
{
    return (int) difftime(fim, inicio);
}
```

```
struct tm* dataHoraAtual()
{
    time_t temp;
    time(&temp);
    return localtime(&temp);
}
```

```
void relógio()
{
    struct tm* tinfo = dataHoraAtual();
    printf("h:%d, m:%d, s:%d \n", tinfo->tm_hour, tinfo->tm_min, tinfo->tm_sec);
}
```

```
void calendario()
{
    struct tm* tinfo = dataHoraAtual();
    printf("dia:%d, mes:%d, ano:%d \n",
        tinfo->tm_mday, tinfo->tm_mon + 1, tinfo->tm_year + 1900);
}
```

Operações numa fila de espera de alunos

Estrutura

```
typedef struct
{
    char nome[50];
    char morada[50];
    int nMec;
    time_t tempoChegada;
    time_t tempoSaida;
}Aluno;
```

```
typedef Aluno Informacao;
```

```
typedef struct nodo
{
    Informacao dados;
    struct nodo* seguinte;
}Nodo;
```

```
typedef struct
{
    Nodo* cabeca;
    Nodo* cauda;
}Fila;
```


Leitura de dados relativos a um Aluno

```
Aluno lerAluno()  
{  
    Aluno aluno;  
    limparTeclado();  
  
    printf("\n Nome? ");  
    gets(aluno.nome);  
  
    printf("\n Morada ?");  
    gets(aluno.morada);  
  
    printf("\n No Mecanografico ?");  
    scanf("%d", &(aluno.nMec));  
  
    return aluno;  
}
```

Visualização de (alguns) dados referentes aos alunos em fila de espera

```
void mostrar(Aluno aluno)
{
    printf("\n  Nome: %s \n  Morada: %s", aluno.nome, aluno.morada);
}
```

```
void listar(Fila fila)
{
    int i = 1;
    Nodo* pNodo = fila.cabeca;
    while (pNodo != NULL)
    {
        printf("\n% da Fila", i++);
        mostrar(pNodo->dados);
        pNodo = pNodo->seguinte;
    }
}
```

Inserção de um Aluno (no fim da fila)

```
void inserir(Aluno aluno, Fila* fila)
{
    Nodo* nodo = (Nodo*)malloc(sizeof(Nodo));
    nodo->dados = aluno;
    nodo->seguinte = NULL;
    nodo->dados.tempoChegada = time(NULL);
    if (fila->cabeca == NULL) // Fila Vazia
    {
        fila->cabeca = nodo;
        fila->cauda = nodo;
    }
    else // O Aluno que chega é inserido no fim da fila
    {
        (fila->cauda)->seguinte = nodo;
        (fila->cauda) = nodo;
    }
}
```

Retirada de um Aluno (do início da fila)

```
Nodo* remover(Fila* fila)
{
    Nodo* rem;
    if (fila->cabeca == NULL) // Verifica se fila Vazia
        return NULL;
    else // Se nao estiver entao retira elemento que
    { // esta à cabeca
        rem = fila->cabeca;
        if (fila->cabeca == fila->cauda)
            // se a fila tem só um elemento, vai ficar vazia
            {
                fila->cabeca = NULL;
                fila->cauda = NULL;
            }
        else //Cabeca passa a ser o anterior 2º elemento
            fila->cabeca = (fila->cabeca)->seguinte;
    }
    rem->dados.temposaida = time(NULL);
    return (rem);
}
```

Determinar o comprimento da fila (número de alunos)

```
int comprimento(Fila fila)
{
    int i = 0;
    Nodo* pNodo = fila.cabeca;
    while(pNodo != NULL)
    {
        pNodo = pNodo->seguinte;
        i++;
    }
    return i;
}
```

Função menu ()

```
int menu()
{
    int opcao;
    fflush(stdin);
    printf("\n\n |-----|");
    printf("\n | 1 - Inserir Aluno |");
    printf("\n | 2 - Retirar Aluno |");
    printf("\n | 3 - Mostrar Alunos na Fila |");
    printf("\n | 4 - Comprimento da Fila |");
    printf("\n | 0 - Sair |");
    printf("\n |-----|\n");
    do
    {
        printf("\nQual a sua opcao ? ");
        scanf("%d", &opcao);
    } while (opcao < 0 || opcao > 4);
    return opcao;
}
```

Função main()

```
void main()
{
    int opcao;
    Fila fila = {NULL, NULL};
    Aluno aluno;
    Nodo* pNode;
    do
    {
        opcao = menu();
        switch (opcao)
        {
            case 1: aluno = lerAluno();
                    inserir(aluno, &fila);
                    break;
            case 2: pNode = remover(&fila);
                    if (pNode != NULL)
                    {
                        printf("\nSaiu da fila o aluno: ");
                        mostrar(pNode->dados);
                        printf("\nEspera na fila: %d",
                            tempoEspera(pNode->dados.tempoSaida, pNode->dados.tempoChegada));
                        free(pNode);
                    }
                    break;
            case 3: listar(fila); break;
            case 4: printf("\n alunos na fila de espera: %d\n", comprimento(fila)); break;
            case 0: exit(EXIT_SUCCESS);
        }
    } while (1);
}
```