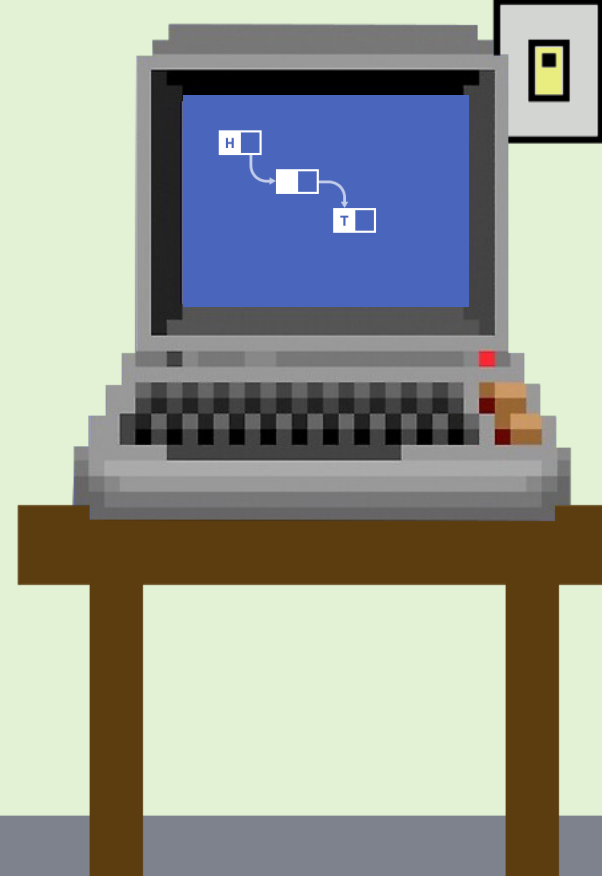
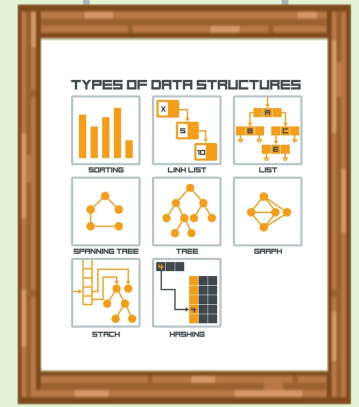


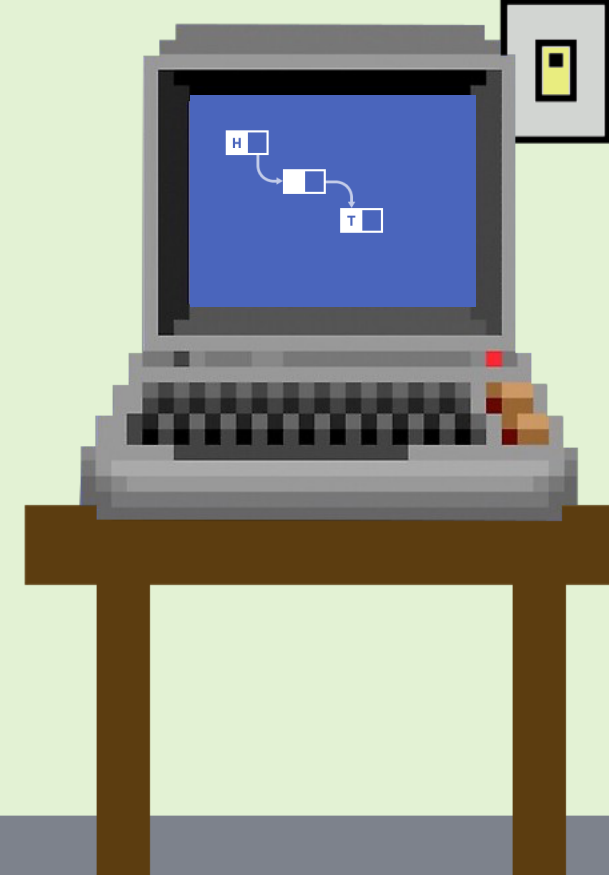
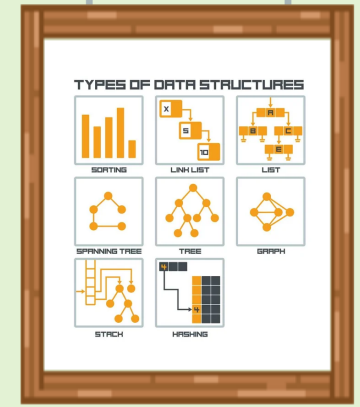
Ponteiros

Estruturas de Dados I



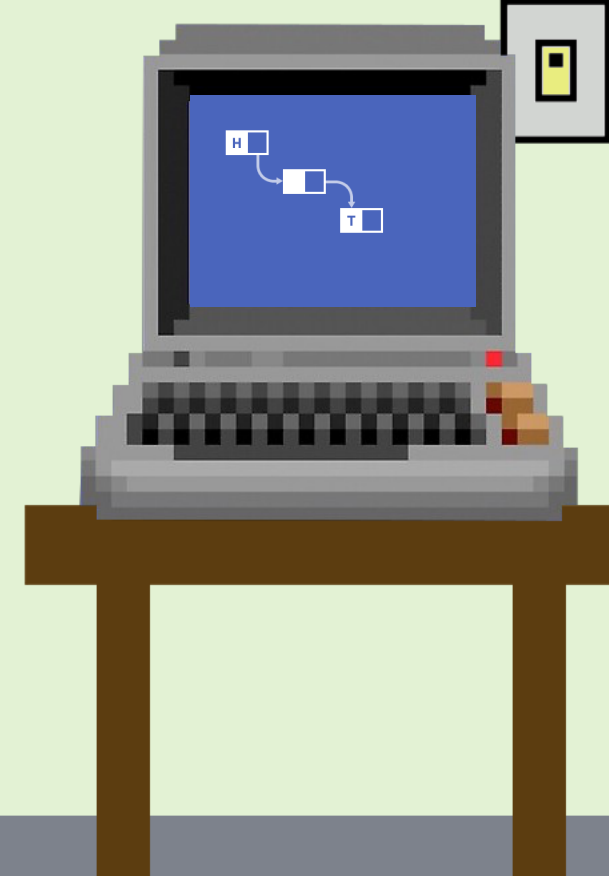
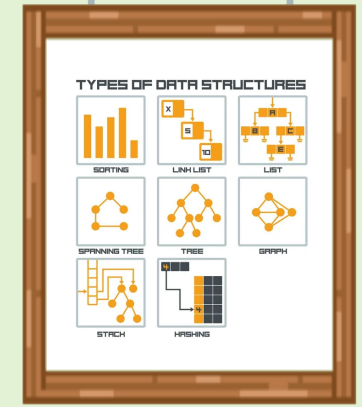
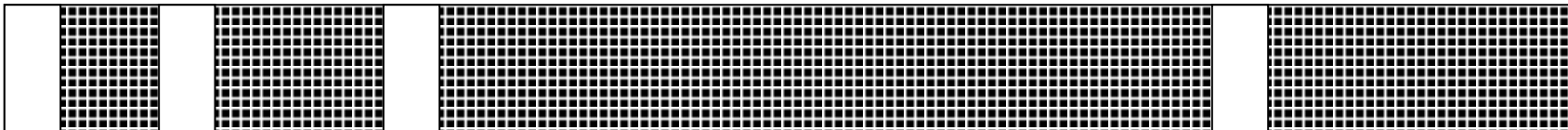
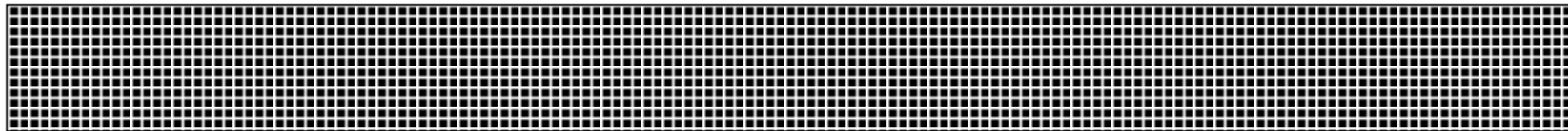
Endereçamento de Memória e Ponteiros

- Fundamentais em qualquer linguagem de programação
- A linguagem C utiliza esses conceitos explicitamente
- Algumas linguagens abstraem em seu uso
 - Mais fáceis
 - Menos recursos
- Cada variável de um programa ocupa um certo número de bytes consecutivos na memória do computador
 - Uma variável do tipo char ocupa 1 byte
 - Uma variável do tipo int ocupa 4 bytes
 - Uma variável do tipo double ocupa 8 bytes
- sizeof - revela quantos bytes uma variável ocupa



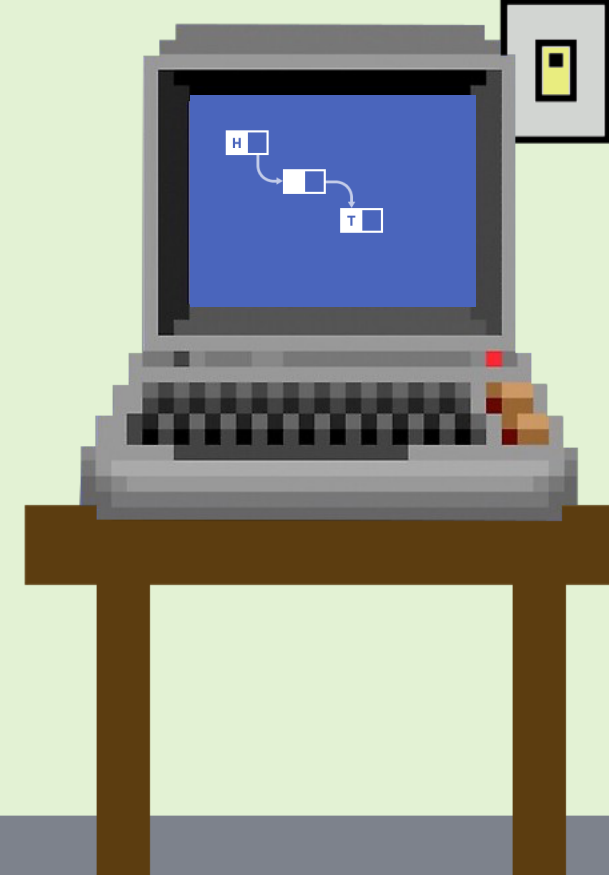
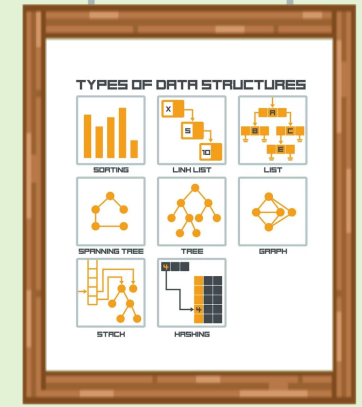
Endereçamento de Memória

- A memória RAM de qualquer computador é uma sequência de bytes
- A posição (0, 1, 2, 3, etc.) que um byte ocupa na sequência é o seu endereço



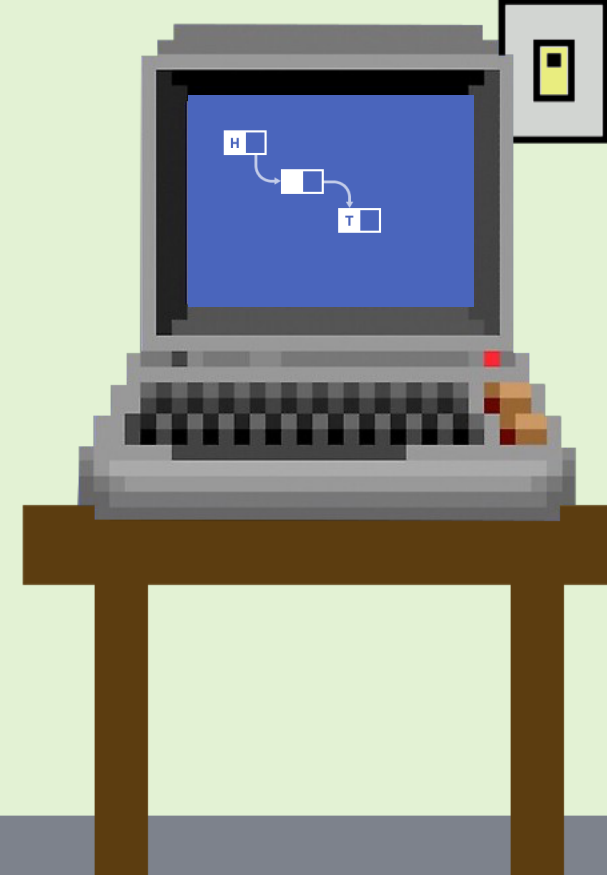
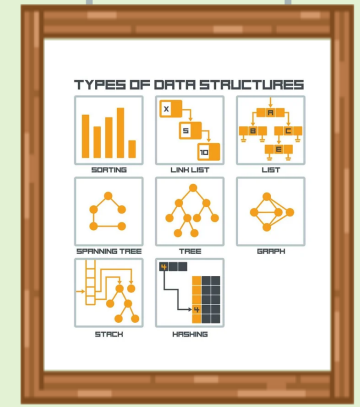
Ponteiros

- Tipo especial de variável que armazena um endereço
- Pode receber o valor NULL (endereço "inválido")
 - Definida na `stdlib.h` e seu valor é geralmente 0
- Se um ponteiro `p` armazena o endereço de uma variável `i`, podemos dizer "p aponta para i"
- Tipos de ponteiros
 - Ponteiros para bytes
 - Ponteiros para inteiros
 - Ponteiros para ponteiros para inteiros
 - Ponteiros para registros
 - Outros



Ponteiros

- Operador & ("endereço de")
 - Aplicado a variáveis
 - Retorna o endereço da posição de memória reservada para variável
- Operador * ("conteúdo de")
 - Aplicado a ponteiros
 - Acessa o conteúdo de memória do endereço armazenado pela variável ponteiro



Ponteiros

```
int a; int* p; int c;
```

```
/* a recebe o valor 5 */  
a = 5;
```

c	-	112
p	-	108
a	5	104

```
/* p recebe o endereço de a  
ou seja, p aponta para a */  
p = &a;
```

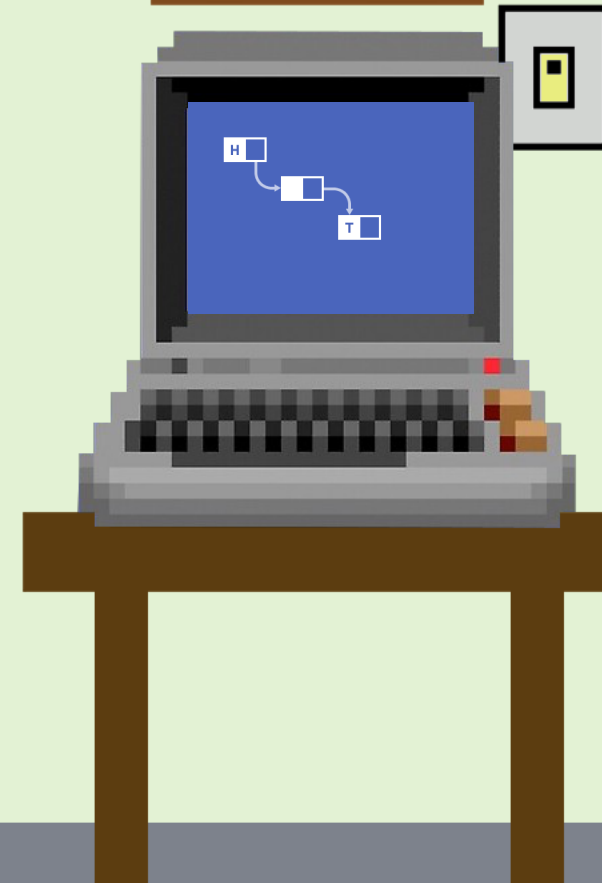
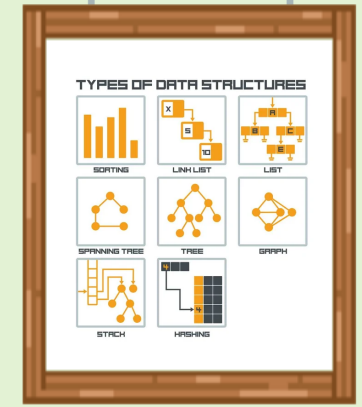
c	-	112
p	104	108
a	5	104

```
/* posição de memória apontada por p  
recebe 6 */  
*p = 6;
```

c	-	112
p	104	108
a	6	104

```
/* c recebe o valor armazenado  
na posição de memória apontada por p */  
c = *p;
```

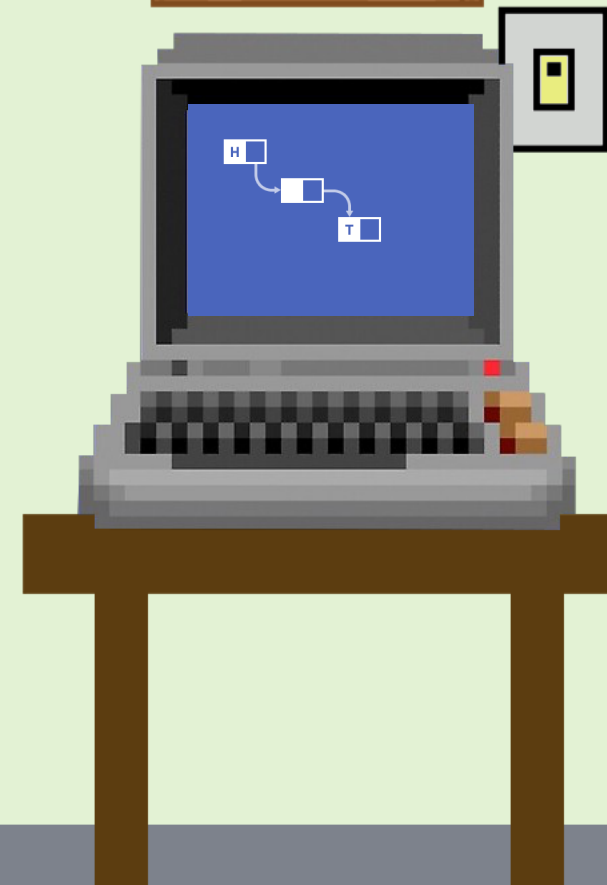
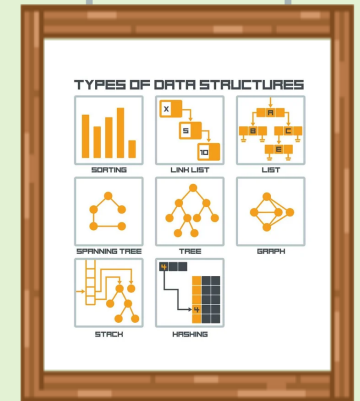
c	6	112
p	104	108
a	6	104



Ponteiros

■ Exemplo 1 - Soma utilizando ponteiros

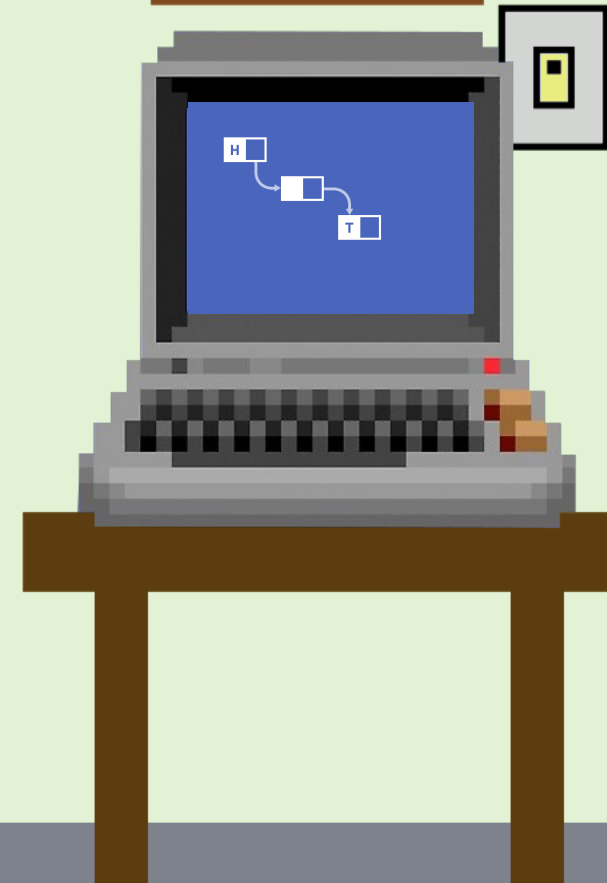
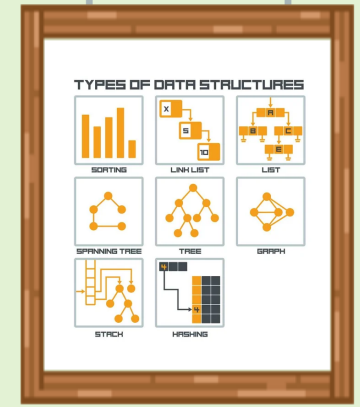
```
int *p;  /// p é um ponteiro para um inteiro
int *q;  /// q é um ponteiro para um inteiro
p = &a;  /// p aponta para a
q = &b;  /// q aponta para b
c = *p + *q;
```



Ponteiros

■ Exemplo 2 - Ponteiro para ponteiro

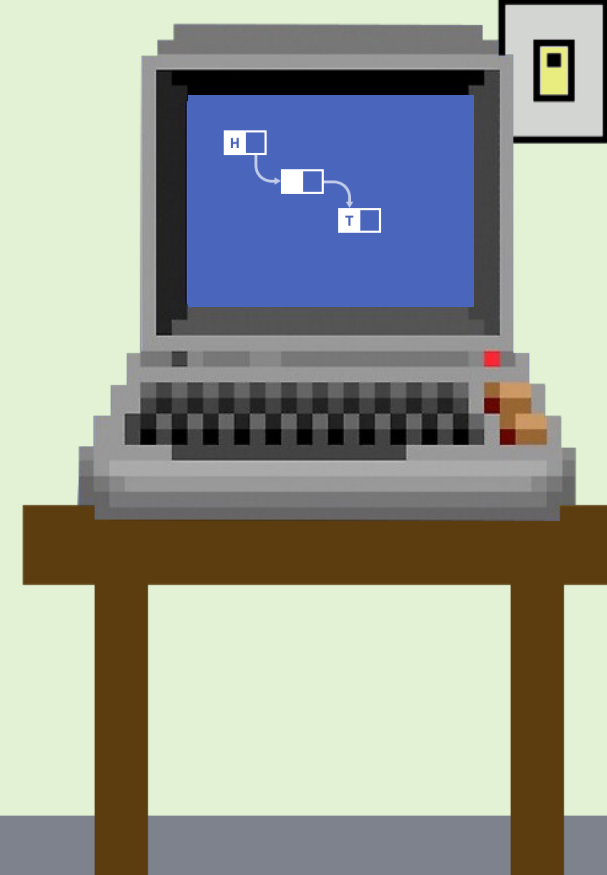
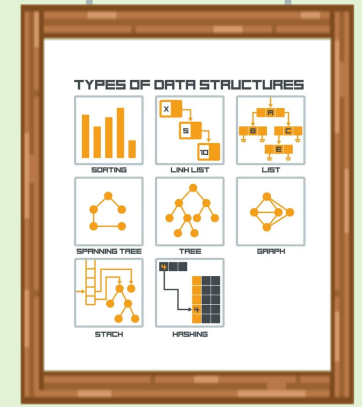
```
int *p; // p é um ponteiro para um inteiro
int **r; // r é um ponteiro para ponteiro para inteiro
p = &a; // p aponta para a
r = &p; // r aponta para p e *r aponta para a
c = **r + b;
```



Ponteiros como parâmetros

- Permitem modificar o valor das variáveis indiretamente
- Possível solução para passagem por referência
- Podem retornar múltiplos valores em uma função
- Podem causar efeitos colaterais
- Exemplo

```
void funcao(int *ptr){  
    *ptr= 3; //altera a variável apontada por ptr  
}
```

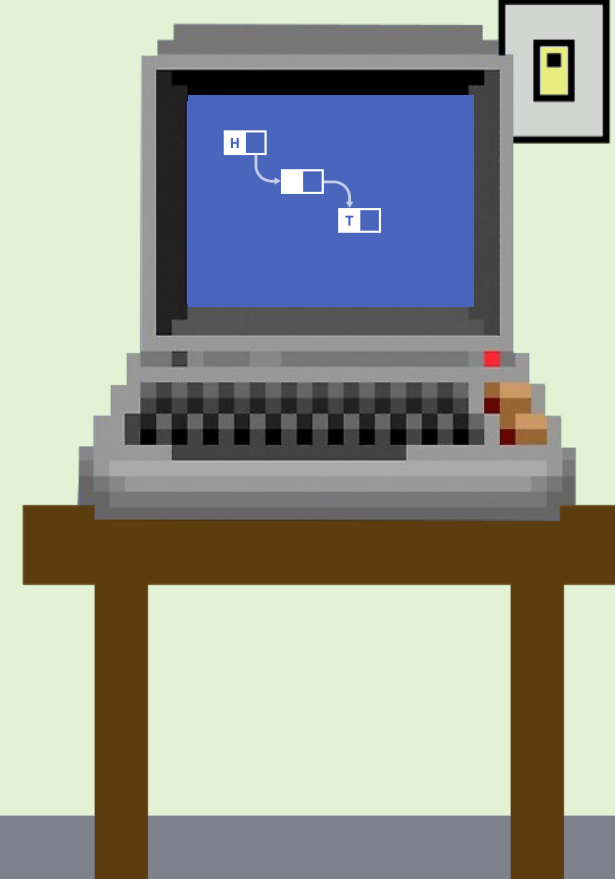
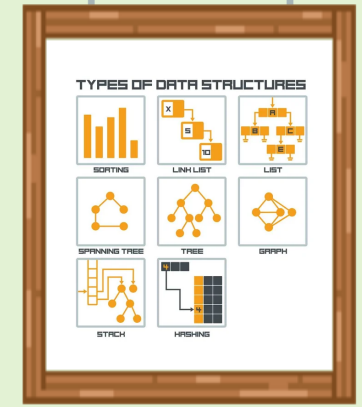


Ponteiros

■ Exemplo 3 - Troca valores de variáveis

```
void troca1 (int i, int j) {  
    int temp;  
    temp = i; i = j; j = temp;  
}
```

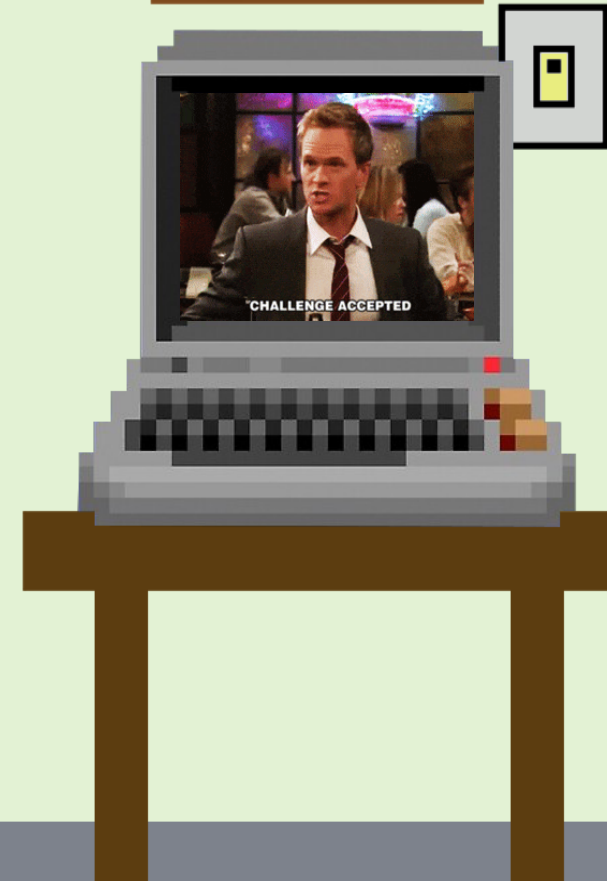
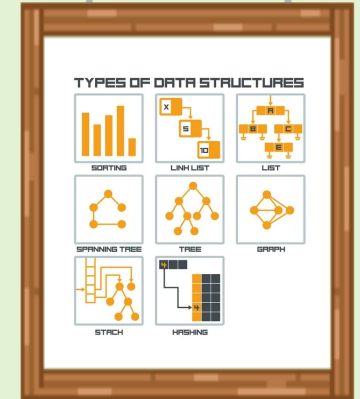
```
void troca2 (int *p, int *q){  
    int temp;  
    temp = *p; *p = *q; *q = temp;  
}
```



Desafio


- Encontre o erro do código abaixo

```
int main (void){  
    int a, b, *p;  
    a = 2;  
    *p = 3;  
    b = a + (*p);  
    printf (~ %d ~, b);  
    return 0;  
}
```



Atividade

■ Resolver a lista de exercícios de revisão

	INSTITUTO FEDERAL Maranhão Campus Caxias	Curso: Ciência da Computação Professor: Luís Fernando Maia	Nota
	Nome: _____		
Disciplina: _____		Data: ____/____/____	

Lista de Exercícios de Revisão

1. Faça um programa que receba o nome e o sexo de uma pessoa, se for do sexo feminino, o programa irá dizer Bem-vinda + nome da pessoa, se for masculino, Bem-vindo + nome da pessoa.
2. Faça um programa que receba um número entre 1 e 9, e mostre as multiplicações dele como apresentado nas tabuadas.
3. Faça um programa que armazena a nota de 10 alunos e mostre qual a maior e a menor nota.
4. Escreva um programa que declare um inteiro, um float e um char, e ponteiros para inteiro, float, e char. Associe as variáveis aos ponteiros (use &). Modifique os valores de cada variável usando os ponteiros. Imprima os valores das variáveis antes e após a modificação.
5. Elaborar um programa que leia dois valores inteiros (A e B). Em seguida faça uma função que retorne a soma do dobro dos dois números lidos. A função deverá armazenar o dobro de A na própria variável A e o dobro de B na própria variável B.
6. Escreva um programa que declare um array de inteiros e um ponteiro para inteiros. Associe o ponteiro ao array. Agora, some mais um (+1) a cada posição do array usando o ponteiro (use *).
7. Faça um programa que leia um quantidade de elementos `p`, crie dinamicamente um vetor de `p` elementos e passe esse vetor para uma função que irá ler seus elementos. Em seguida, o vetor preenchido deve ser impresso.
8. Escreva um programa em linguagem C que solicita do usuário a quantidade de alunos de uma turma e aloca um vetor de notas (números reais). Depois de ler as notas, o programa deve imprimir a média aritmética da turma. Obs: não deve ocorrer desperdício de memória; e após ser utilizada a memória deve ser devolvida.

