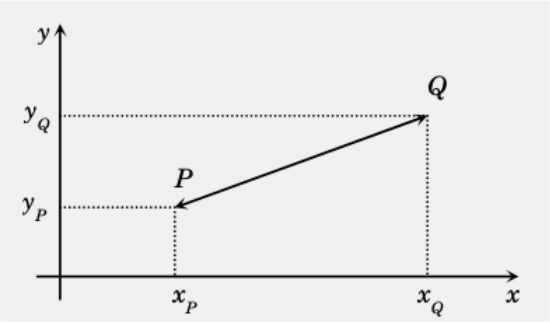
Lista 8 - Estruturas e Ponteiros

1. Codifique um programa para criar uma variável de tipo anônimo, capaz de armazenar o título, o autor, a editora e o ano de publicação de um livro; atribuir valores aos seus campos e exibi-la no vídeo.
2. Defina um tipo de estrutura rotulada para representar números complexos da forma `a+b·i`, sendo a **a** parte real e **b** a imaginária. Em seguida, crie uma função para calcular a soma de dois números complexos e codifique um programa para testar seu funcionamento. Use atribuição para inicializar os campos membros das variáveis.
3. Defina um tipo de estrutura nomeada para representar pontos no plano através de suas coordenadas cartesianas. Em seguida, crie uma função para calcular a distância entre dois pontos e codifique um programa para testar seu funcionamento. Use a função `scanf()` para inicializar os campos membros das variáveis com valores lidos do teclado.

   Dica: a distância entre dois pontos P e Q, conforme ilustração ao lado, é dada pela seguinte fórmula:

$$\overline{PQ} = \sqrt{(x_Q - x_P)^2 + (y_Q - y_P)^2}$$



4. Usando o tipo definido no exercício anterior, para armazenamento de pontos do plano cartesiano, defina um tipo de estrutura para representar segmentos de retas através dos seus extremos. Crie uma função para deter- minar o comprimento de um tal segmento e faça um programa para testá-la.

5. Defina um tipo de estrutura para armazenar um horário composto de hora, minutos e segundos. Crie e inicialize uma variável desse tipo e, em seguida, mostre seu valor no vídeo usando o formato "99:99:99".
6. Defina um tipo de estrutura para armazenar os dados de um vôo como, por exemplo, os nomes das cidades de origem e destino, datas e horários de partida e chegada. Crie uma variável desse tipo e atribua valores aos seus membros usando a notação de ponto e, depois, inicialização. Usando o tipo de estrutura, crie e inicialize uma tabela com os dados de todos os vôos de um aeroporto e codifique uma rotina para exibi-la em vídeo.
7. Defina um tipo de estrutura, para representar livros, contendo título, autor e ano de publicação. Em seguida, codifique uma função para preencher uma tal estrutura com dados obtidos via teclado. Codifique uma função para preencher uma tabela cujos elementos são estruturas representando livros.
8. Calculadora Polimórfica com Ponteiros de Funções. Crie um programa que simule uma calculadora básica que pode realizar operações diferentes (adição, subtração, multiplicação e divisão) usando **ponteiros de função**. O usuário escolhe a operação que deseja realizar e o programa chama a função correspondente.
9. Sistema de Impressão de Mensagens Personalizadas. Crie um programa que possa imprimir diferentes mensagens personalizadas com base em diferentes cenários, como uma mensagem de boas-vindas, uma mensagem de erro e uma mensagem de despedida. Use **ponteiros de função** para selecionar a mensagem correta.
10. Sistema de Ordenação Polimórfico. Desenvolva um programa que possa ordenar um vetor de inteiros de diferentes maneiras (ordem crescente, ordem decrescente e ordem absoluta) usando **ponteiros de função**.
11. Sistema de Manipulação de Strings com Operações Polimórficas. Crie um programa que permita manipular uma string de diferentes maneiras (converter para maiúsculas, minúsculas e inverter a string) usando **ponteiros de função.**
12. Função que Retorna um Ponteiro para o Maior Elemento de um Array. Escreva uma função chamada `encontraMaior` que recebe um array de inteiros e seu tamanho, e retorna um ponteiro para o maior elemento do array.

```
int* encontraMaior(int* array, int tamanho);
```

```
//exemplo de uso
int arr[] = {5, 3, 9, 2, 10};
int* maior = encontraMaior(arr, 5);
printf("O maior valor é %d\n", *maior);
```

13. Função que Retorna um Ponteiro para uma Substring. Escreva uma função chamada `substring` que recebe uma string, um índice inicial e um comprimento, e retorna um ponteiro para uma nova string que contém a substring especificada.

```
char* substring(const char* str, int inicio, int comprim
ento);

//exemplo de uso
char* str = "OpenAI";
char* sub = substring(str, 2, 3);
printf("Substring: %s\n", sub);
free(sub);  // Liberar a memória da substring
```

14. Função que Retorna o Endereço de um Elemento em uma Matriz. Crie uma função chamada `obtemElemento` que recebe uma matriz (`int** matriz`), o número de linhas, o número de colunas, um índice de linha e um índice de coluna. A função deve retornar um ponteiro para o elemento na posição especificada.

```
int* obtemElemento(int** matriz, int linhas, int coluna
s, int linha, int coluna);

//Exemplo de uso
int linhas = 3, colunas = 3;
int** matriz = (int**)malloc(linhas * sizeof(int*));
for (int i = 0; i < linhas; i++) {
    matriz[i] = (int*)malloc(colunas * sizeof(int));
}
matriz[1][1] = 42;
int* elemento = obtemElemento(matriz, linhas, colunas,
1, 1);
if (elemento != NULL) {
```

```
    printf("Elemento encontrado: %d\n", *elemento);
} else {
    printf("Índices fora dos limites.\n");
}
// Liberar a memória alocada
for (int i = 0; i < linhas; i++) {
    free(matriz[i]);
}
free(matriz);
```

15. A biblioteca padrão da linguagem C oferece uma função polimórfica para ordenação de vetores cujo protótipo é o seguinte:

    `void qsort(void *v, int n, int t, int (*c)(const void *,const void *));`
    sendo v um ponteiro para o vetor a ser ordenado, `n` o número de elementos no vetor, `t` o tamanho em bytes de cada elemento e c um ponteiro para a função de comparação a ser usada durante ordenação. A novidade nesse protótipo é o tipo `void*`, que serve para declarar ponteiros capazes de receber qualquer tipo de endereço. Isso quer dizer que, por exemplo, o parâmetro `v` pode receber como valor inicial tanto o endereço de um vetor de caracteres quanto o endereço de um vetor de números reais ou ainda um vetor de estruturas. O único problema com ponteiros void é que não é permitido o acesso a dados a partir deles e, portanto, temos que usar casts com eles. Com base nessas informações, crie um vetor de estruturas e tente ordená-lo por cada um de seus campos, um de cada vez, usando a função `qsort()`.