RELATÓRIO - ESTRUTURA DE DADOS.
Caio Marcos Moita Mourão || Semestre 2

Estrutura do Programa

O arquivo `necio_trabalho.c` implementa operações estatísticas sobre um vetor de inteiros criado dinamicamente, utilizando ponteiros para manipulação eficiente dos dados.

1. Criação e Inicialização do Vetor

- Alocação dinâmica:
  - Função `numbers_create(int size)` aloca um vetor de inteiros com tamanho definido em tempo de execução, usando `malloc`. Retorna um ponteiro para o início do vetor, permitindo acesso e manipulação dos elementos via ponteiro.
- Preenchimento:
  - Função `numbers_read(int *numbers, int size)` preenche o vetor com valores aleatórios de 0 a 99, usando um laço. O ponteiro permite modificar diretamente o vetor alocado na memória.

2. Exibição dos Valores

- Função `numbers_show`:
  - Imprime todos os valores do vetor em formato de lista, facilitando a visualização dos dados gerados.

3. Operações Estatísticas

- Média: `numbers_average(int *numbers, int size)` percorre o vetor, soma os elementos e retorna a média.
- Maior valor: `maior_num(int *numbers, int size)` retorna o maior elemento do vetor.
- Menor valor: `menor_num(int *numbers, int size)` retorna o menor elemento do vetor.
- Números pares: `num_pares(int *numbers, int size)` conta e exibe os pares.
- Números ímpares: `num_impares(int *numbers, int size)` conta e exibe os ímpares.
- Múltiplos de cinco: `multiplos_de_cinco(int *numbers, int size)` conta e exibe os múltiplos de cinco.

Todas essas funções recebem o vetor por ponteiro, acessam seus elementos e retornam estatísticas ou imprimem resultados.

4. Liberação de Memória

- Função `numbers_destroy(int *numbers)`:
    - Libera o espaço do vetor criado, evitando desperdício de memória.

Uso dos Ponteiros

Todas as operações principais usam ponteiros para acessar e modificar o vetor. Ao passar `numbers` (do tipo `int *`) para uma função, ela acessa e altera o vetor original alocado, sem criar cópias desnecessárias. Isso torna o programa eficiente e permite manipulação direta dos dados na memória.

Fluxo no `main`

1. Gera tamanho aleatório para o vetor.
2. Cria o vetor dinamicamente usando ponteiro.
3. Preenche o vetor com valores aleatórios.
4. Exibe o vetor.
5. Calcula média, maior, menor, pares, ímpares e múltiplos de 5.
6. Exibe todos os resultados.
7. Libera a memória usada pelo vetor.

Trecho das Funções e Explicações

Criação do vetor

c

```c
int *numbers_create(int size) {
  int *num = NULL;
  num = (int *)malloc(size * sizeof(int));
  if (num == NULL)
    return NULL;
  return num;
}
```

- Explicação: Aloca dinamicamente um vetor de inteiros. Se a alocação falhar, retorna `NULL`. O ponteiro `num` aponta para o início do vetor, permitindo acesso aos elementos.

Preenchimento do vetor

c

```c
void numbers_read(int *numbers, int size) {
  if (numbers != NULL) {
    for (int i = 0; i < size; i++)
      numbers[i] = rand() % 100;
  }
}
```

- Explicação: Preenche o vetor com valores aleatórios de 0 a 99. O ponteiro permite modificar diretamente o vetor original.

Cálculo da média

c

```c
float numbers_average(int *numbers, int size) {
  float sum = 0.0;
  if (numbers != NULL)
    for (int i = 0; i < size; i++)
      sum += numbers[i];
  if (sum != 0)
    return sum / (float)size;
  return 0;
}
```

- Explicação: Soma todos os elementos do vetor e retorna a média. O ponteiro permite acessar cada elemento.

Maior valor

```c
int maior_num(int *numbers, int size){
  int maior = numbers[0];
  for (int i = 1; i < size; i++) {
    if (numbers[i] > maior) {
      maior = numbers[i];
    }
  }
  return maior;
}
```

- Explicação: Percorre o vetor e retorna o maior valor encontrado.

Menor valor

```c
int menor_num(int *numbers, int size){
  int menor = numbers[0];
  for (int i = 1; i < size; i++) {
    if (numbers[i] < menor) {
      menor = numbers[i];
    }
  }
  return menor;
}
```

- Explicação: Percorre o vetor e retorna o menor valor encontrado.

Números pares

c

```c
int num_pares(int *numbers, int size){
  int pares = 0;
  printf("Numeros Pares: [");
  for(int i = 0; i < size; i++){
    if(numbers[i] % 2 == 0){
      printf("%d ", numbers[i]);
      pares++;
    }
  }
  printf("]\n");
  return pares;
}
```

Explicação: Conta e exibe os números pares do vetor.

Números ímpares

c

```c
int num_impares(int *numbers, int size){
  int impares = 0;
  printf("Numeros Impares: [");
  for(int i = 0; i < size; i++){
    if(numbers[i] % 2 != 0){
      printf("%d ", numbers[i]);
      impares++;
    }
  }
  printf("]\n");
  return impares;
}
```

- Explicação: Conta e exibe os números ímpares do vetor.

Múltiplos de cinco

```c
int multiplos_de_cinco(int *numbers, int size){
  int multiplos = 0;
  printf("Multiplos de 5: [");
  for (int i = 0; i < size; i++) {
    if (numbers[i] % 5 == 0) {
      printf("%d ", numbers[i]);
      multiplos++;
    }
  }
  printf("]\n");
  return multiplos;
}
```

- Explicação: Conta e exibe os múltiplos de cinco do vetor.

Exibição dos valores

```c
void numbers_show(int *numbers, int size) {
  printf("Todos os numeros: [");
  if (numbers != NULL) {
    for (int i = 0; i < size; i++)
      if (i == (size - 1))
        printf("%d", numbers[i]);
      else
        printf("%d, ", numbers[i]);
  }
  printf("]\n");
}
```

- Explicação: Imprime todos os valores do vetor em formato de lista.

Endereço do Repositório

RockyzFX/Ponteiros-AlocacaoDinamica

Arquivo analisado: `necio_trabalho.c`