Algoritmos e Estrutura de Dados II

Exercício Prático
Bubblesort, Insertionsort e Selectionsort

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso
Instituto de Engenharia

Agenda

Roteiro da Aula

Exercício 1

Desenvolva um programa que gere um vetor com valores aleatórios.

Exercício 1

Primeiramente, vamos definir algumas constantes:

- `tamanho_vetor`: define a quantidade de elementos no vetor.
- `valor_max`: define o maior valor a ser inserido no vetor.
- `valor_min`: define o menor valor a ser inserido no vetor.

Definindo as constantes, respectivamente, com os valores 10, 20 e 0, o código na linguagem C fica assim:

```
1 #define tamanho_vetor 10
2 #define valor_max 20
3 #define valor_min 0
```

Exercício 1

Agora, dentro da função `main()` vamos alocar um vetor de inteiros de tamanho igual `tamanho_vetor`:

```
1 int *vetor;
2 vetor = malloc(tamanho_vetor*sizeof(int));
```

Não se esqueça de desalocar no final:

```
1 free(vetor);
```

Exercício 1

- Para inserir números aleatórios no vetor utilizamos a função `rand()`.
- Acesse esse link para entender o funcionamento da função `rand()` e `srand()`.
- A função `rand()` é frequentemente usada em conjunto com a função `time(NULL)`, da seguinte forma: `srand(time(NULL))`.

Exercício 1

Vetor Aleatório

Vamos inserir valores aleatórios no vetor:

```
srand(0);
for (i = 0; i < tamanho_vetor; i++)
    vetor[i] = (rand() % valor_max) + valor_min;
```

Em seguida, vamor imprimí-los:

```
for(i = 0; i < tamanho_vetor; i++)
    printf("%d ", vetor[i]);
```

Execute o código utilizando `srand(time(NULL))`, lembre-se que é necessário incluir a biblioteca `<time.h>`.

Exercício 1

Solução

O código completo fica:

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define tamanho_vetor 10
#define valor_max 20
#define valor_min 0
int main() {
    int i, *vetor;
    srand(time(NULL));
    vetor = malloc(tamanho_vetor*sizeof(int));
    for (i = 0; i < tamanho_vetor; i++)
        vetor[i] = (rand() % valor_max) + valor_min;
    for(i = 0; i < tamanho_vetor; i++)
        printf("%d ", vetor[i]);
    free (vetor);
    return 0;
}
```

Roteiro da Aula

Exercício 2

Bubblesort

Dado o pseudocódigo a seguir, implemente o algoritmo de ordenação **Bubblesort** e execute-o em um vetor formado por valores aleatórios.

Exercício 2

Pseudocódigo

Algoritmo 1: Bubblesort

Entrada: Vetor $V[0..n-1]$, tamanho n
Saída: Vetor V ordenado

```
1 início
2     para (i ← 1 até n − 1) faça
3         para (j ← 0 até n − i − 1) faça
4             se (V[j] > V[j + 1]) então
5                 Trocar V[j] ↔ V[j + 1]
```

Exercício 2

Solução

```
1  void bubblesort(int n, int *vetor) {
2      int i, j, aux;
3      for(i=1; i<n; i++) {
4          for(j=0; j<n-i; j++) {
5              if(vetor[j]>vetor[j+1]) {
6                  aux = vetor[j+1];
7                  vetor[j+1] = vetor[j];
8                  vetor[j] = aux;
9              }
10         }
11     }
12 }
```

Roteiro da Aula

Exercício 3

Selectionsort

Dado o pseudocódigo a seguir, implemente o algoritmo de ordenação **Selectionsort** e execute-o em um vetor formado por valores aleatórios.

Exercício 3

Pseudocódigo

Algoritmo 2: Selectionsort

Entrada: Vetor $V[0..n-1]$, tamanho n
Saída: Vetor V ordenado

```
1 início
2     para i ← 0 até n − 2 faça
3         min ← i
4         para j ← i + 1 até n − 1 faça
5             se V[j] < V[min] então
6                 min ← j
7         trocar V[min] ↔ V[i]
```

Exercício 3

Solução

```
1  void selectionsort(int n, int *vetor) {
2      int i, j, aux, min;
3      for(i = 0; i < n-1; i++) {
4          min = i;
5          for(j = i+1; j < n; j++) {
6              if(vetor[j] < vetor[min])  {
7                  min = j;
8              }
9          }
10         aux = vetor[min];
11         vetor[min] = vetor[i];
12         vetor[i] = aux;
13     }
14 }
```

Roteiro da Aula

Exercício 4

Inseriionsort

Dado o pseudocódigo a seguir, implemente o algoritmo de ordenação **Inserionsort** e execute-o em um vetor formado por valores aleatórios.

Exercício 4

Pseudocódigo

Algoritmo 3: Insertionsort

Entrada: Vetor $V[0..n-1]$, tamanho n
Saída: Vetor V ordenado

```
1 início
2     para (i ← 1 até n − 1) faça
3         chave ← V[i]
4         j ← i − 1
5         enquanto (j ≥ 0 AND V[j] > chave) faça
6             V[j + 1] ← V[j]
7             j ← j − 1
8         V[j + 1] ← chave
```

Exercício 4

Solução

```
1  void insertionsort(int n, int *vetor) {
2      int i, j, chave;
3      for(i = 1; i < n; i++) {
4          chave = vetor[i];
5          j = i - 1;
6          while(j>=0 && vetor[j]>chave) {
7              vetor[j+1] = vetor[j];
8              j = j - 1;
9          }
10         vetor[j+1] = chave;
11     }
12 }
```

Roteiro da Aula

Exercício 5

Tempo Processamento

- Calcule o tempo de processamento para ordenar um vetor contendo mil valores utilzando cada um dos algoritmos de ordenação. Para isso, utilizamos a função `clock()`.
- Acesse esse link para entender o funcionamento da função `clock()`.

Introdução

Analise o exemplo a seguir que demostra como deve-se medir o tempo de processamento de um determinado programa.

Exercício 5

Tempo Processamento

```c
#include <time.h>
int main () {
    clock_t tempoInicial, tempoFinal;
    double tempoGasto;
    tempoInicial = clock(); // Obtem o clock inicial
    /* Adicione aqui a chamada do algoritmo de ordenacao */
    tempoFinal = clock(); // Obtem o clock final
    // Calcula o tempo gasto
    tempoGasto = (tempoFinal-tempoInicial)*1000 / CLOCKS_PER_SEC);
    printf("Tempo em milisegundos: %lf", (double)tempoGasto);
}
```

Roteiro da Aula

Exercício 6

Ordenando um Registro

- Execute o algoritmo **Bubblesort**, entretanto, dessa vez deverá ordenar um vetor formado por registros, confome a estrutura de dados a seguir.
- Em seguida, calcule seu tempo de processamento para ordenacão de um vetor de dados aleatórios contendo mil valores.

Exercício 6

Pseudocódigo

Algoritmo 4: Selectionsort

Entrada: Vetor $V[0..n-1]$, tamanho n
Saída: Vetor V ordenado

```
1 início
2     para i ← 0 até n − 2 faça
3         min ← i
4         para j ← i + 1 até n − 1 faça
5             se V[j] < V[min] então
6                 min ← j
7         trocar V[min] ↔ V[i]
```

Exercício 6

Estrutura de Dados

```
1 typedef struct {
2     int chave;
3     int valor;
4 } dado;
```

Exercício 6

Ordenando um Registro

O primeiro passo é criar um vetor de registros aleatórios.

Exercício 6

Algoritmo de Ordenacão

```c
#define tamanho_vetor 10
#define valor_max 20
#define valor_min 0
int main () {
    srand(0);
    vetor = malloc(tamanho_vetor*sizeof(dado));
    for (int i = 0; i < tamanho_vetor; i++) {
        vetor[i].chave = (rand() % valor_max) + valor_min;
        vetor[i].valor = i;
    }
    free(vetor);
}
```

Exercício 6

Ordenando um Registro

- Agora vamos adaptar o algoritmo de ordenacão **Bubblesort**.
- O ponto chave é alterar as linhas 4 e 5.

Exercício 2

Pseudocódigo

Algoritmo 5: Bubblesort

Entrada: Vetor $V[0..n-1]$, tamanho n
Saída: Vetor V ordenado

```
1 início
2     para (i ← 1 até n − 1) faça
3         para (j ← 0 até n − i − 1) faça
4             se (V[j] > V[j + 1]) então
5                 Trocar V[j] ↔ V[j + 1]
```

Exercício 6

Estrutura de Dados

```
1  void bubblesort(int n, dado *vetor) {
2      int i, j;
3      dado aux;
4      for(i=1; i<n; i++) {
5          for(j=0; j<n-i; j++) {
6              if(vetor[j].chave > vetor[j+1].chave)  {
7                  aux.chave = vetor[j+1].chave ;
8                  aux.valor = vetor[j+1].valor ;
9                  vetor[j+1].chave = vetor[j].chave;
10                 vetor[j+1].valor = vetor[j].valor;
11                 vetor[j].chave = aux.chave;
12                 vetor[j].valor = aux.valor;
13             }
14         }
15     }
16 }
```

Exercício 6

Ordenando um Registro

Voce pode criar uma funcão que troque os elementos.

Exercício 6

Funcão Tocar

```
1 void trocar(dado *d1, dado *d2) {
2   dado aux;
3   aux.chave = d2->chave;
4   aux.valor = d2->valor;
5   d2->chave = d1->chave;
6   d2->valor = d1->valor;
7   d1->chave = aux.chave;
8   d1->valor = aux.valor;
9 }
```

Exercício 6

Bubblesort

```
1  void bubblesort(int n, dado *vetor) {
2      int i, j;
3      for(i=1; i<n; i++) {
4          for(j=0; j<n-i; j++) {
5              if(vetor[j].chave > vetor[j+1].chave)  {
6                  trocar(&vetor[j], &vetor[j+1]);
7              }
8          }
9      }
10 }
```

Exercício 6

Ordenando um Registro

Por fim, vamos adicionar as variáveis para contabilizar o tempo gasto.

Exercício 6

Bubblesort

```
clock_t tempoInicial, tempoFinal;
dado *vetor;
double tempoGasto;
tempoInicial = clock(); // Obtem o clock inicial
bubblesort(tamanho_vetor, vetor);
tempoFinal = clock(); // Obtem o clock final
// Calcula o tempo gasto
tempoGasto = (tempoFinal-tempoInicial)*1000 / CLOCKS_PER_SEC;
```

Algoritmos e Estrutura de Dados II

Exercício Prático
Bubblesort, Insertionsort e Selectionsort

prof. Frederico Santos de Oliveira

Universidade Federal de Mato Grosso
Instituto de Engenharia