```
// importações;
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <locale.h>
//declaração de esc -> variavel global para auxilio da opção 9 da ordenação de vetores, que, para funcionar corretamente, precisa saber qual forma de dados o
//usuario escolheu.
int esc = 0;
//relatorios (Declaraçoes);
void relatorio (int comp, int cont);
void relatorio (int comp, int cont, int lac1);
void relatorio (int comp, int cont, int lac1, int lac2);
void relatorio (int comp, int cont, int lac1, int lac2, int lac3);
void relatorioShell (int comp, int cont, int lac1, int lac2, int lac3);
//Trabalhadores (Declaraçoes);
void pop_ordm (int*vetor,int tam);
void pop_semi (int*vetor,int tam);
void pop_deso (int*vetor,int tam);
void mostrar (int*vet, int tam);
void zerar (int*vetor, int tam);
//Buscadores(Declaraçoes);
void busca_Binaria (int *vetor, int tam);
void busca_Sequencial(int*vetor, int tam);
//Auxiliadores (Declaraçoes);
int semi();
int dinamico();
int menu();
void vetOrdem(int*vetor, int tam);
// Ordenadores (Declaraçoes);
```

```
void bubblesort(int*vetor,int tam);

void shellSort(int*vetor,int tam);

void selectionSort(int*vetor,int tam);

void quickSort(int*vetor, int left, int right);

void mergeSort();

void insertionSort(int*vetor,int tam);

void Merge(int *vetor, int ini, int meio, int fim, int *vetAux);

void Sort(int *vetor, int inicio, int fim, int *vetAux);

void MergeSort(int *vetor, int tam);

void qkSt(int*vetor, int left, int right); // <- ordenador especifico para uso da função de semi-ordenação!

//inicio.

int main() {

int gat = 0; // como o nome sugere, "gat" é o gatilho para 'navegaçao/verificavel de escolha' dos Switch;

int tam = 6000; // valor de tam, define o tamanho do vetor no caso do "FIXO";

//iniciando o vetor;

int vetor[tam]; // criação do vetor, usando o tam como tamanho (tam logicamente e int, o vetor? um vetor de inteiros.);

//iniciando controladores de contagem de tempo (padronizar e agilizar);

int inicio,fim,tempo; //(Declaração das variaveis de controle de tempo;

/*

Apresentação do trabalho, de forma resumida, com os nomes dos e intregantes do grupo, o curso e o semestre;

*/

printf("\nBem vindo ao trabalho de APS da turma de Ciencia da computacao - Quarto semestre. \n\n");

printf("Materia: Estrutura de dados.\n\n");

printf("Nomes: \nArnon.\nKamilah.\nGabriel.\nVinicius.\nWagner.\n\n");

system("\n\n\n pause");

system("cls");

do{ //começo da implementação de estrutura de escolhas;

/*
```

No começo tinhamos um grande problema de implementar os Switch's como estrutura de escolha, sem o devido tratamento

eles se tornam uma fonte confiavel de bugs e erros no codigo, como meta de primeira etapa de refinamento,

implementamos testes logicos em alguns pontos para restringir o usuario a escolhas dentro da proposta do programa

assim, em um menu com "Escola [1], [2] ou [3] o ussuario não sera capaz de escolhar 27 e continuar normalmente.

a propria aplicação, o indicara que este numero não e aceito, e mostrara novamente a ultima tela/menu que

o ussuario estava;

*/

//começo da aplicação...;

printf("----------");

printf("\n -- Escolha seu vetor --\n");

printf("----------\n");

printf("Qual vetor de inteiros deseja? \n\n[1] - Fixo.\n[2] - Dinamico.\n\n\n----------\n[0] - Sair\n\n"); // primeira escolha do programa;

scanf("%i",&gat); // usando o gat para a primeira escolha;

// sobre escolhas por switch.;

/*

As melhores para agirem como menus, claro, se tratadas cuidadosamente, e bem aplicadas;

*/

switch (gat) //usando valor do 'gat' para escolha;

{

case 1: // vetor fixo, nada mais faz que, pegar o valor de tam declarado na sua criação e usalo (puro) como tamanho do vetor;

zerar(vetor,tam);

/*

Aqui, passamos por parametro o vetor e o seu tamanho, para a função que, percorre o vetor, setando o valor 0 em

cada posição, motivo disso? eliminar lixo de memoria. podiamos simplesmente chamar a proxima função tambem

ela ja colocaria os valores aleatorios em cada posição, assim, subistituindo o lixo ou valores ja existentes,

maaasss... zerar, e a forma mais segura de evitar erros, e preparar o vetor para uso;

*/

vetOrdem(vetor,tam); // chamdndo o metodo que, dara ao ussuario a liberdade de escolher qual tipo de dados deseja dentro do seu vetor;

if (tam < 5000) // em cima!! vetOrdem como teste...

{ //Teste logico para exibição de vetores...

//quando criamos um vetor muito longo, podemos perder minutos so "imprimindo" o mesmo apos sua criação

//para evitar isso, foi retirada a obrigatoriedade; o ussuario so imprimira seu vetor se quiser por meio da opção "Observar vetor";

//isto deixa as coisas mais dinamicas;

mostrar(vetor,tam); // aqui, temos uma função simples composta de prnt's.f's - para informações, de uma olhada na função;

system("pause");

system("cls");

} else {

system("cls");

printf("----------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Criacao --\n");

printf("----------\n");

printf("\nPor questoes de dinamica, nao mostraremos vetores acima de cinco mil (5000) posicoes \nmas se mesmo assim desejar ver seu vetor, escolha a opcao 'Observar' a seguir. \n");

printf("----------");

system("\n\npause"); // pausando a aplicação, para que o usuario possa ver oque esta acontecendo;

system("cls");

}

gat = 0;

break;

```
//-----------------------------------------------------------------------------------------

case 2:

tam = 0;

// parte do vetor dinamico...

/*

Aqui, damos ao ussuario a escolha de selecionar o tamanho do tam, assim, escolhendo o tamanho do vetor com qual

o mesmo queira interagir, logicamente, não se trata de um vetor dinamico, esta mais para

um vetor customizavel.

*/

tam = (int)dinamico(); // para evitar algum tipo de erro, convertemos o valor recebido da função dinamico em int;

vetor[tam]; // vetor recebe tam, com o tamanho especifico;

zerar(vetor,tam); // zera o mesmo;

vetOrdem(vetor,tam); // e pergunta ao ussuario qual tipo de dados o mesmo deseja;

if (tam < 5000)

{

mostrar(vetor,tam); // aqui, temos uma função simples composta de prnt's.f's - para informações, de uma olhada na função;

system("pause");

system("cls");

} else {

system("cls");

printf("-----------------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Criacao --\n");

printf("-----------------------------------------------------------------------------------------------------------------------\n");

printf("\nPor questoes de dinamica, nao mostraremos vetores acima de cinco mil (5000) posicoes \nmas se mesmo assim desejar ver seu vetor, escolha a opcao 'Observar' a seguir. \n");

printf("-----------------------------------------------------------------------------------------------------------------------");
```

```
system("\n\npause"); // pausando a aplicação, para que o usuario possa ver oque esta acontecendo;

system("cls");

}

gat = 0;

break;

//-----------------------------------------------------------------------------------------

case 0:

exit(1); // Opção para fechar a aplicaçao;

break;

//-----------------------------------------------------------------------------------------

default: // Aqui, casso o usuario tente digitar uma opção difente das dadas pela aplicação, a mensagem abaixo sera exibida, e o codigo sera repetido Pelo While.

printf("Opcao invalida.");

system("Pause");

system("cls");

break;

//-----------------------------------------------------------------------------------------

}

}while(gat != 0);

do{

// Proximo passo da interaçõa com o ussuario, aqui, mostranmos para que nosso programa foi feito! (literalmente :V)

gat = 0;

printf("-----------------------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao --\n");

printf("-----------------------------------------------------------------------------------------------------------------------------\n");

printf("\nO que deseja fazer agora? \n\n\n[1] Organizar vetor. \n\n[2] Observar vetor. \n\n[3] Buscar algum valor.\n\n\n----------\n[0] - Sair\n\n");

scanf("%i", &gat);

switch (gat) //usando valor do 'gat' para escolha.
```

```
{

case 1:

goto protect; // usando goto para avançar a aplicacao;

break;

//-------------------------------------------------------------------------------------------

case 2:

system("cls");

printf("-------------------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Observador --\n");

printf("-------------------------------------------------------------------------------------------------------------------------\n\n");

mostrar(vetor,tam); // chamando a função para imprimir o vetor, passamos para o mesmo, o proprio vetor, e seu tamanho;

system("pause");

system("cls");

break;

//-------------------------------------------------------------------------------------------

case 3: // Opçoes de busca no vetor;

system("cls");

printf("-------------------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Buscador --\n");

printf("-------------------------------------------------------------------------------------------------------------------------\n\n");

printf("Deseja qual tipo de busca?\n\n");

printf("\n[1] - Binaria. \n[2] - Sequencial.\n");

scanf("%i", &gat);

switch (gat)

{

case 1:

busca_Binaria(vetor,tam); // opção binaria, (mais infos no metodo);
```

```
break;
case 2:
busca_Sequencial(vetor,tam); // opção sequencial, (mais infos no metodo);
break;
default:
printf("\nOpcao invalida. ");
system("Pause");
system("cls");
}
break;
//-----------------------------------------------------------------------------------------------
case 0:
exit(1);
break;
//-----------------------------------------------------------------------------------------------
default:
printf("Opcao invalida, ");
system("Pause");
system("cls");
break;
//-----------------------------------------------------------------------------------------------
}
}while(gat != 0);
gat = 0; // zerando o gat... importante para o uso dessa "tecnica" de usar ele como cursor e sempre zera-lo apos o uso.
do{
protect:
system("cls"); // limpando o console.
printf("-----------------------------------------------------------------------------------------------------------------");
printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Ordenadores e Tempos --\n");
```

```
printf("-------------------------------------------------------------------------------------------------------------------------\n");

printf("\nEscolha o metodo...? \n\n\n[1] - Bubble.\n[2] - Selection.\n[3] - Shell.\n[4] - Quick.\n[5] - insertion.\n[6] - Merge.\n\n");

printf("\n\n\n\n\n\n\n-------------------------------------------------------------------------------------------------------------------------\n");

printf("Caso queira organizar com todas as formas disponiveis aperte [9]\n\n");

printf("\n\n----------------------------------------------------------\n[0] - Sair");

printf("\n----------------------------------------------------------\n");

printf("\nLembrando que seu vetor possui %i posicoes.",tam); // mostrando quantas possiçoes o vetor possui.

printf("\n\n----------------------------------------------------------\n");

switch (esc) // mostrando qual tipo de dados foi escolhido;

{

case 1:

printf("\nSeus tipos de dados sao Ordenados\n");

break;

case 2:

printf("\nSeus tipos de dados sao Semi-ordenados\n");

break;

case 3:

printf("\nSeus tipos de dados sao Desordenados\n");

break;

}

scanf("%i",&gat);

switch (gat)

{

//-------------------------------------------------------------------------------------------------------------------------

case 1:

system("cls");

printf("-------------------------------------------------------------------------------------------------------------------------");
```

```
printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Ordenadores e Tempos > Bubble Sort --\n");

printf("-------------------------------------------------------------------------------------------------------------------------\n");

inicio = GetTickCount(); // inicio do cronometro

bubblesort(vetor,tam); // chamando o metodo de ordenaçao Bubblesort

fim = GetTickCount(); // fim do cronometro;

mostrar(vetor,tam); // imprimindo o vetor.

tempo = fim-inicio;

printf("\n\nVetor organizado com sucesso...\n");

printf("\n\nTempo decorrido: %d milisegundos",tempo);

system("pause");

system("cls");

break;

//-------------------------------------------------------------------------------------------------------------

case 2:

system("cls");

printf("-------------------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Ordenadores e Tempos > Selection Sort --\n");

printf("-------------------------------------------------------------------------------------------------------------------------\n");

inicio = GetTickCount();

selectionSort(vetor,tam);

fim = GetTickCount();

mostrar(vetor,tam);

tempo = fim-inicio;

printf("\n\nVetor organizado com sucesso...\n");

printf("\n\nTempo decorrido: %d milisegundos",tempo);

system("pause");

system("cls");

break;
```

```
//-----------------------------------------------------------------------------------------------------------

case 3:

system("cls");

printf("-----------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Ordenadores e Tempos > Shell Sort --\n");

printf("-----------------------------------------------------------------------------------------------------------------\n");

inicio = GetTickCount();

shellSort(vetor,tam);

fim = GetTickCount();

mostrar(vetor,tam);

tempo = fim-inicio;

printf("\n\nVetor organizado com sucesso...\n");

printf("\n\nTempo decorrido: %d milisegundos",tempo);

system("pause");

system("cls");

break;

//-----------------------------------------------------------------------------------------------------------

case 4:

system("cls");

printf("-----------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Ordenadores e Tempos > Quick Sort --\n");

printf("-----------------------------------------------------------------------------------------------------------------\n");

inicio = GetTickCount();

quickSort(vetor,0,tam-1);

fim = GetTickCount();

mostrar(vetor,tam);

tempo = fim-inicio;
```

```
printf("\n\nVetor organizado com sucesso...\n");

printf("\n\nTempo decorrido: %d milisegundos",tempo);

system("pause");

system("cls");

break;

//-----------------------------------------------------------------------------------------------------------

case 5:

system("cls");

printf("-----------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Ordenadores e Tempos > Insertion Sort --\n");

printf("-----------------------------------------------------------------------------------------------------------------\n");

inicio = GetTickCount();

insertionSort(vetor,tam);

fim = GetTickCount();

mostrar(vetor,tam);

tempo = fim-inicio;

printf("\n\nVetor organizado com sucesso...\n");

printf("\n\nTempo decorrido: %d milisegundos",tempo);

system("pause");

system("cls");

break;

//-----------------------------------------------------------------------------------------------------------

case 6:

system("cls");

printf("-----------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Ordenadores e Tempos > Merge Sort --\n");

printf("-----------------------------------------------------------------------------------------------------------------\n");
```

```
inicio = GetTickCount();

MergeSort(vetor,tam);

fim = GetTickCount();

mostrar(vetor,tam);

tempo = fim-inicio;

printf("\n\nVetor organizado com sucesso...\n");

printf("\n\nTempo decorrido: %d milisegundos",tempo);

system("pause");

system("cls");

break;

//-----------------------------------------------------------------------------------------------------------

case 9: // chamando todos os metodos de organização em cadeia

system("cls");

printf("-----------------------------------------------------------------------------------------------------------------
--------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao >
Ordenadores e Tempos > Tudo em Ordem --\n");

printf("-----------------------------------------------------------------------------------------------------------------
--------\n");

printf ("\nNesta parte, temos todos os metodos trabalhando em sequencia. Leia com atencao
o texto abaixo:");

printf("\n-------------------------------------------------------------\n");

printf("\n\nPara garantir um total funcionamento de cada metodo de ordenacao usamos o
vetor escolhido por voce, de forma sequencial \nem um processo que consiste em zera-lo,
repopula-lo de forma ordenada /semi-ordenada /ou desordenada (conforme
escolhido)\ncronometra-lo e imprimir para entao repetir o processo com o proximo metodo
de ordenacao.");

printf("\n\nAssim cada vetor trabalha em igualdade de condicoes.");

printf("\n-------------------------------------------------------------\n");

system("pause");

system("cls"); // limpando o console.

printf("-----------------------------------------------------------------------------------------------------------------
--------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao >
Ordenadores e Tempos > Tudo em Ordem --\n");
```

```
printf("-----------------------------------------------------------------------------------------------------------------\n");

// -----------Organizando pelo BubbleSort (Usando vetor ja criado)-----------------------------------------

printf("\n\n\n-------------------------------------------------------------\n");

printf("\n\nVetor organizado com sucesso (Metodo BubbleSort)...\n");

switch (esc)

{

case 1:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_ordm(vetor,tam); // re-populamos de forma aleatoria

break;

case 2:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_semi(vetor,tam); // re-populamos de forma semi-ordenada

break;

case 3:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_deso(vetor,tam); // re-populamos de forma desordenada

break;

}

// que poderiam haver nele, estando ordenados ou não

// Na sequencia re-populamos ele.

inicio = GetTickCount(); // chamada inicial do cronometro.

bubblesort(vetor,tam); // metodo de organização.

fim = GetTickCount(); // chamada final do vetor.

tempo = fim-inicio;

printf("\n\nTempo decorrido: %d milisegundos",tempo);

// ------------Organizando pelo SelectionSort (Zerando o vetor ja organizado, e, re-preenchendo-o)----------
```

```
printf("\n\n\n------------------------------------------------------------\n");

printf("\n\nVetor organizado com sucesso (Metodo SelectionSort)...\n");

switch (esc)

{

case 1:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_ordm(vetor,tam); // re-populamos de forma aleatoria

break;

case 2:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_semi(vetor,tam); // re-populamos de forma semi-ordenada

break;

case 3:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_deso(vetor,tam); // re-populamos de forma desordenada

break;

}

inicio = GetTickCount();

selectionSort(vetor,tam);

fim = GetTickCount();

tempo = fim-inicio;

printf("\n\nTempo decorrido: %d milisegundos",tempo);

// ------------Organizando pelo ShellSort (Zerando o vetor ja organizado, e, re-preenchendo-o)----------------

printf("\n\n\n------------------------------------------------------------\n");

printf("\n\nVetor organizado com sucesso (Metodo ShellSort)...\n");

switch (esc)

{

case 1:
```

```
zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_ordm(vetor,tam); // re-populamos de forma aleatoria

break;

case 2:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_semi(vetor,tam); // re-populamos de forma semi-ordenada

break;

case 3:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_deso(vetor,tam); // re-populamos de forma desordenada

break;

}

inicio = GetTickCount();

shellSort(vetor,tam);

fim = GetTickCount();

tempo = fim-inicio;

printf("\n\nTempo decorrido: %d milisegundos",tempo);

// ------------Organizando pelo QuicklSort (Zerando o vetor ja organizado, e, re-preenchendo-o)-------------------

printf("\n\n\n------------------------------------------------------------\n");

printf("\n\nVetor organizado com sucesso (Metodo QuickSort)...\n");

switch (esc)

{

case 1:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_ordm(vetor,tam); // re-populamos de forma aleatoria

break;

case 2:
```

```
zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_semi(vetor,tam); // re-populamos de forma semi-ordenada

break;

case 3:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_deso(vetor,tam); // re-populamos de forma desordenada

break;

}

inicio = GetTickCount();

quickSort(vetor,0,tam-1);

fim = GetTickCount();

tempo = fim-inicio;

printf("\n\nTempo decorrido: %d milisegundos",tempo);

// ------------Organizando pelo QuicklSort (Zerando o vetor ja organizado, e, re-preenchendo-o)-------------------

printf("\n\n\n------------------------------------------------------------\n");

printf("\n\nVetor organizado com sucesso (Metodo insertionSort)...\n");

switch (esc)

{

case 1:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_ordm(vetor,tam); // re-populamos de forma aleatoria

break;

case 2:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_semi(vetor,tam); // re-populamos de forma semi-ordenada

break;

case 3:
```

```
zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_deso(vetor,tam); // re-populamos de forma desordenada

break;

}

inicio = GetTickCount();

insertionSort(vetor,tam);

fim = GetTickCount();

tempo = fim-inicio;

printf("\n\nTempo decorrido: %d milisegundos",tempo);

// ------------Organizando pelo MergeSort (Zerando o vetor ja organizado, e, re-preenchendo-o)-------------------

printf("\n\n\n------------------------------------------------------------\n");

printf("\n\nVetor organizado com sucesso (Metodo mergeSort)...\n");

switch (esc)

{

case 1:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_ordm(vetor,tam); // re-populamos de forma aleatoria

break;

case 2:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_semi(vetor,tam); // re-populamos de forma semi-ordenada

break;

case 3:

zerar(vetor,tam); // Primeiro zeramos o vetor, assim eliminamos qualquer sequencia de numeros

pop_deso(vetor,tam); // re-populamos de forma desordenada

break;

}

inicio = GetTickCount();
```

```
MergeSort(vetor,tam);

fim = GetTickCount();

tempo = fim-inicio;

printf("\n\nTempo decorrido: %d milisegundos\n\n",tempo);

system("\n\npause");

break;

//-----------------------------------------------------------------------------------------------------------

case 0: // Opcao para sair da aplicação

exit(1);

break;

//-----------------------------------------------------------------------------------------------------------

default :

printf("\nOpcao invalida.\n");

system("\n\npause");

system("cls");

goto protect;

break;

}

} while (gat != 0);

}

/*

Relatorios são usados diretamente e em conjunto com os metodos de ordeção, tanto que, somente eles os chamam, aqui, temos, 5 metodos

mas, como ja visto em POO, cada qual possui uma assinatura, porque disso?

cada metodo de ordenação e unico, alguns simples, emglobam apenas um laço, uma troca, e uma serie do comparações

outros... aaah... outros não... e e por isso que temos cada qual com "comp" (Comparaçoes) e "cont" (trocas) como parametros fixos,

e as unicas coisas que almentão são os parametros responsaveis por contar as voltas de cada laço.

Mais datalhes do funcionamento no BubbleSort.

*/
```

```
void relatorio (int comp, int cont)
{
printf("\n\nNumero de trocas: %i",cont);
printf("\n\nNumero de comparacoes: %i\n\n",comp);
}
void relatorio (int comp, int cont, int lac1)
{
printf("\n\nNumero de trocas: %i",cont);
printf("\n\nNumero de comparacoes: %i",comp);
printf("\n\nNumero de voltas do primeiro laco: %i\n\n",lac1);
}
void relatorio (int comp, int cont, int lac1, int lac2)
{
printf("\n\nNumero de trocas: %i",cont);
printf("\n\nNumero de comparacoes: %i",comp);
printf("\n\nNumero de voltas do primeiro laco: %i",lac1);
printf("\n\nNumero de voltas do segundo laco: %i\n\n",lac2);
}
void relatorio (int comp, int cont, int lac1, int lac2, int lac3)
{
printf("\n\nNumero de trocas: %i",cont);
printf("\n\nNumero de comparacoes: %i",comp);
printf("\n\nNumero de voltas do primeiro laço: %i",lac1);
printf("\n\nNumero de voltas do segundo laço: %i",lac2);
printf("\n\nNumero de voltas do terceiro laço: %i\n\n",lac3);
}
void relatorioShell (int comp, int cont, int lac1, int lac2, int lac3) //<-teste
{
printf("\n\nNumero de trocas: %i",cont);
printf("\n\nNumero de comparacoes: %i",comp);
printf("\n\nNumero de Quebras do vetor: %i",lac1);
```

```
printf("\n\nNumero de voltas do primeiro laco: %i",lac2);

printf("\n\nNumero de Trocas no pivo: %i\n\n",lac3);

}

//--------------------------------------------------------------------------------

// Ordenadores.

void bubblesort(int*vetor,int tam)

{

int trocas = 0, lac1 = 0, lac2 = 0, comp = 0; // declaração local das variaveis usadas para os trabalhos do relatorio;

int aux = 0;

for (int a = 0 ; a < tam; a++, lac1++){ // aqui, implementamos o laço, dentro dele, nossa primeira variavel de contagem que, a cada volta do mesmo

// sera implementada em +1;

for (int b = a + 1 ; b < tam ; b++, comp++, lac2++){ // contagem do segundo laco;

if(vetor[a] > vetor[b]){ // momento da troca de valores do metodo booble

aux = vetor[a];

vetor[a] = vetor[b];

vetor[b] = aux;

trocas++; // tambem e o momento que a variavel "Trocas" e implementada +1;

}

}

}

relatorio(comp,trocas,lac1,lac2); // chamando o relatorio, e passando por referencia os valores acumulados duranta a execusao do bubbleSort

}

//--------------------------------------------------------------------------------

void shellSort(int*vetor,int tam) // Shell simples...

{

int trocas = 0, lac1 = 0, lac2 = 0, lac3 = 0, comp = 0; // declaração dos contadores

int i , j , value; // declaracão dos valores usados no vetor

int gap = 1; // auxiliar

do {
```

```
gap = 3*gap+1;
lac1++;
} while(gap < tam);
do {
gap /= 3;
for(i = gap; i < tam; i++, lac2++) {
value = vetor[i];
j = i - gap;
lac3++;
while (comp++, j >= 0 && value < vetor[j]) {
vetor[j + gap] = vetor[j];
j -= gap;
trocas ++;
}
vetor[j + gap] = value;
}
}while(gap > 1);
relatorioShell(comp,trocas,lac1,lac2,lac3);
}
//--------------------------------------------------------------------------------
void selectionSort(int*vetor,int tam)
{
int trocas = 0, lac1 = 0, lac2 = 0, comp = 0;
int i, j, k, tmp, tr;
for(i = 0; i < tam-1; i++, comp++, lac1++)
{
tr = 0;
k = i;
tmp = vetor[i];
for(j = i+1; j < tam; j++, comp++, lac2++)
{
```

```
if(vetor[j] < tmp)
{
k = j;
tmp = vetor[j];
tr = 1;
}
}
if(tr)
{
vetor[k] = vetor[i];
vetor[i] = tmp;
trocas++;
}
}
relatorio(comp,trocas,lac1,lac2);
}
//---------------------------------------------------------------------------
void quickSort(int*vetor, int left, int right)
// (vetor,0,tam-1) (o 0 é passado para o left de inicio,
//porque e e novamente chamado, e ai, o left tera outro valor)
{
// quick tem um serio problema com o metodo de relatorio atual... ele e recursivo, significado? ele se chama, e isso faz com que a precisão do relatorio (Ja duvidosa) va para outra galaxia...
int trocas = 0, lac1 = 0, lac2 = 0, lac3 = 0, comp = 0;
int i = 0, j = 0, x = 0, y = 0;
int teste;
i = left;
j = right;
x = vetor[(left + right) / 2];
while(i <= j) {
while(vetor[i] < x && i < right) {
```

```
i++;
}
while(vetor[j] > x && j > left) {
j--;
}
if(i <= j) {
y = vetor[i];
vetor[i] = vetor[j];
vetor[j] = y;
i++;
j--;
teste += trocas++;
}
}
if(j > left) {
quickSort(vetor, left, j);
}
if(i < right) {
quickSort(vetor, i, right);
}
}
//--------------------------------------------------------------------------------
// O metodo de ordenação Merge-sort e recursivo, aqui vemos ele dividido em 3 funçoes, onde cada uma chama a seguinte;
// 1ª
void Merge(int *vetor, int ini, int meio, int fim, int *vetAux)
{
int esq = ini;
int dir = meio;
for (int i = ini; i < fim; ++i) {
if ((esq < meio) and ((dir >= fim) or (vetor[esq] < vetor[dir]))) {
```

```
vetAux[i] = vetor[esq];
++esq;
}
else {
vetAux[i] = vetor[dir];
++dir;
}
}
//copiando o vetor de volta
for (int i = ini; i < fim; ++i) {
vetor[i] = vetAux[i];
}
}
//2ª
void Sort(int *vetor, int inicio, int fim, int *vetAux)
{
if ((fim - inicio) < 2) return;
int meio = ((inicio + fim)/2);
Sort(vetor, inicio, meio, vetAux);
Sort(vetor, meio, fim, vetAux);
Merge(vetor, inicio, meio, fim, vetAux);
}
//3ª
void MergeSort(int *vetor, int tam) //função que o usuario realmente chama
{
//criando vetor auxiliar
int vetAux[tam];
Sort(vetor, 0, tam, vetAux);
}
//--------------------------------------------------------------------------------
void insertionSort(int*vetor,int tam)
```

```
{
int trocas = 0, lac1 = 0, lac2 = 0, lac3 = 0, comp = 0;
int i, j, tmp;
for(i = 1; i < tam; i++)
{
tmp = vetor[i];
for(j = i-1; j >= 0 && tmp < vetor[j]; j--)
{
vetor[j+1] = vetor[j];
}
vetor[j+1] = tmp;
}
}
//--------------------------------------------------------------------------------
void qkSt(int*vetor, int left, int right) //ordenador Customizado para uso da função de semi-ordenação!
{
int i = 0, x = 0, y = 0;
int j = (right / 2);
int rg = j;
int lf = left;
i = left;
x = vetor[(lf + rg) / 2];
while(i <= j) {
while(vetor[i] < x && i < rg) {
i++;
}
while(vetor[j] > x && j > lf) {
j--;
}
if(i <= j) {
```

```
y = vetor[i];
vetor[i] = vetor[j];
vetor[j] = y;
i++;
j--;
}
}
if(j > lf) {
quickSort(vetor, lf, j);
}
if(i < rg) {
quickSort(vetor, i, rg);
}
}
//--------------------------------------------------------------------------------
//Trabalhadores
//Gerando os numeros aleatorios e ja colocando-os no vetor;
void pop_deso (int*vetor,int tam)
{
for (int i = 0 ; i < tam ; i++)
{
vetor[i] = rand () % tam; // aqui temos o nosso vetor sendo percorrido de i ate tam (valor total do vetor), a coda avanço,
// "rand" gera um numero aleatorio que vai de 0 a tam, oque significa?
// se o vetor tem 100 posiçoes, teremos numeros aleatorio de 0 a 100, se forem 2000... numeros de 0 a 2000;
}
}
void pop_semi (int*vetor,int tam)
{
for (int i = 0 ; i < tam ; i++)
{
```

vetor[i] = rand () % tam; // aqui temos o nosso vetor sendo percorrido de i ate tam (valor total do vetor), a coda avanço,

// "rand" gera um numero aleatorio que vai de 0 a tam, oque significa?

// se o vetor tem 100 posiçoes, teremos numeros aleatorio de 0 a 100, se forem 2000... numeros de 0 a 2000;

}

qkSt(vetor,0,tam);

}

void pop_ordm (int*vetor,int tam)

{

for (int i = 0 ; i < tam ; i++)

{

vetor[i] = rand () % tam; // aqui temos o nosso vetor sendo percorrido de i ate tam (valor total do vetor), a coda avanço,

// "rand" gera um numero aleatorio que vai de 0 a tam, oque significa?

// se o vetor tem 100 posiçoes, teremos numeros aleatorio de 0 a 100, se forem 2000... numeros de 0 a 2000;

}

quickSort(vetor,0,tam);

}

// so para imprimir o vetor;

void mostrar (int*vetor,int tam){

// sobre...

printf("_____\n"); // aqui temos o começo da impressao, importante que, como podem notar esta parte esta fora do laço, afinal, so sera impressa 1 vez

printf("| Posicao || valor ||\n");

printf("------------------------\n");

for (int i = 0 ; i < tam ; i++){

printf("| || |\n", i, vetor[i]);

printf("| %4i %5d |\n", i, vetor[i]); // impressão dos valores, esses numeros antes de "I" e "D", deixam o valor fixos... como assim?

// Ex: reservamos para impressão 4 espaços e depois mais 5 | _ _ _ _ , _ _ _ _ _ | enquanto nossa

// nossa sequencia numerica não estourar esse espaço, tudo que estiver na frente das reservas

```
// não sera deslocado para frente nem para trás, muito util para criação de mini tabelas...
}
printf("_____|\n");
}
// para lidar de forma simples (Talvez não a mais indicada) como o "lixo" de memoria;
void zerar (int*vetor,int tam)
{
int vtm = sizeof(vetor);
for(int i = 0 ; i < vtm; i++)
{
vetor[i] = 0;
}
}
//--------------------------------------------------------------------------------
//Buscadores.
// Busca Binaria no vetor.
void busca_Binaria (int*vetor,int tam)
{
bool achou = false;
int inf = 0; // limite inferior (o primeiro índice de vetor em C é zero )
int sup = tam-1; // limite superior (termina em um número a menos. 0 a 9 são 10 números)
int meio; // divisão dos valores.
int var = 0; //numero buscado.
int aux; // auxiliar.
system("cls");
printf("------------------------------------------------------------------------------------------------------------------------");
printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Buscador > Busca Binaria --\n");
printf("------------------------------------------------------------------------------------------------------------------------\n");
```

```
printf("\n\nDevemos lembra-los que para a realizacao de busca binaria em vetores, o mesmo precisa estar obrigatoriamente ordenado. \nPor este motivo, caso o mesmo nao tenha sido previamente ordenado, tomamos a liberdade de ordena-lo automaticamente a seguir.\n\n\n\n");

printf("-------------------------------------------------------------------------------------------------------------------------\n");

system("pause");

system("cls"); // primeiro ordenamo...

for (int a = 0 ; a < tam; a++){ // primeiro laço, vai de 0 a tam

for (int b = a + 1 ; b < tam ; b++){ // segundo laço percorre o vetor de a+1 ate quando o tam for menor que b

if(vetor[a] > vetor[b]){ // se o valor na posicao A for menos que o na B

aux = vetor[a]; //trocam.

vetor[a] = vetor[b];

vetor[b] = aux;

}

}

}

printf("-------------------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Buscador > Busca Binaria --\n");

printf("-------------------------------------------------------------------------------------------------------------------------\n");

printf ("Digite o valor que deseja buscar\n\n");

scanf("%i",&var); // guardando o valor escolhido pelo usuario;

while (inf <= sup) // primeiro laco while, pega o valor a esquerda do vetor, Ex: 0 , o valor a direita Ex: 15, e somaos, para

{ // para obter o tamanho do vetorr.

meio = (inf + sup)/2; // logo apos, divide o mesmo por 2;

if (var == vetor[meio]) // se o valor pedido pelo ussuario for exatamente o do meio, ja de primeira veremos a msg abaixo.

printf("\nValor desejado localizado | Posicao: %i | Valor: %i |\n\n\n",meio ,vetor[meio], achou = true);
```

```
if (var < vetor[meio]) //caso não, e o valor pedido for menor que o do meio, o codigo diminui o valor do meio.

sup = meio-1; //Ex vetor[meio] = 5; o valor pedido foi 3... meio que estava na posição 4, agora diminuira 1, indo para a 3;

else

inf = meio+1;

}

if (achou == false)

{

printf("-------------------------------------------------------------------------------------------------------------------------\n");

printf("Valor nao encontrado\n");

}

system("pause");

system("cls");

}

// Busca Sequencial no vetor.

void busca_Sequencial(int*vetor, int tam) // busca sequencial... e feita de forma parecida com a ordenadora Bublle sort.

{ // nossa aplicacao percorrera todo o vetor, de posicao em posicao conferindo os valores,

int var; // toda vez que achar nosso valor, contara;

bool achou = false; // variavel booleana de controle

system("cls"); // limpando o console

printf("-------------------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados > Opcoes de interacao > Buscador > Busca Sequencial --\n");

printf("-------------------------------------------------------------------------------------------------------------------------\n");

printf ("Digite o valor que deseja buscar\n\n");

scanf("%i",&var); // lendo o valor a ser buscado

for (int i = 0 ; i < tam ; i++){ // laco responsavel por correr o vetor

if (vetor[i] == var) // condicao : vetor na posição I e igual o valor digitado?
```

```
printf("\nValor desejado localizado em: |Posicao: %i |\n\n\n", i+1, vetor[i],achou = true); // caso seja, é imprimido na tela, e a bool achou

} // se torna TRUE

if(achou == false) // caso ele percorra todo o vetro d e não ache o valor solicitado;

{

printf("-------------------------------------------------------------------------------------------------------------------------\n");

printf("\nValor nao localizado.\n\n");

}

system("Pause");

system("cls");

}

//-----------------------------------------------------------------------------

//Auxiliadores

int dinamico()

{ // responsavel pelo nosso vetor "dinamico"

nozero:

int temp, valor; // declaramos as variaveis que vamos usar;

system("cls");

printf("-------------------------------------------------------------------------------------------------------------------------\n");

printf("\n -- Escolha seu vetor > Dinamico --\n");

printf("-------------------------------------------------------------------------------------------------------------------------\n");

printf("Nesta parte, voce tem a liberdade de escolher um tamanho especifico para o seu vetor de inteiros.\n");

printf("\nQual tamanho deseja?\n");

scanf("%i",&temp); // temp armazena o valor que o usuario deseja para o tamanho do seu vetor;

if (temp <= 0)

{

printf("\nPor Favor digite algum valor aceitavel para o seu vetor.\n");

system("pause\n\n");
```

```
system("cls");

goto nozero;

}

int *test; // processo para verificar se o computador possui memoria para o valor solicitado;

test = (int*)malloc(temp*sizeof(int)); // tentativa de reserva

if(test!=NULL)

{

valor = temp; // caso de tudo certo, temos nossa variavel valor com o tamanho escolhedo pelo usuario

}else{

printf("Memoria insuficiente; hahaha!"); // caso não aja memoria, esta menssagem aparecera para o usuario;

system("pause");

exit(1); // fechando a aplicação;

}

return valor;

}

void vetOrdem(int*vetor, int tam){ // Função para escolher a forma dos dados. (ordenado / semi-ordenado / desordenado);

nd:

int *p; // declaração do ponteiro;

int gat = 0;

system("cls");

printf("-----------------------------------------------------------------------------------------------------------------");

printf("\n -- Escolha seu vetor > Tipo de vetor > Ordem dos dados --\n");

printf("-----------------------------------------------------------------------------------------------------------------\n");

printf("\nDeseja seus dados em qual estado?");

printf("\n\n\n [1] - Ordenado.\n\n [2] - Semi-Ordenado.\n\n [3] - Desordenado.\n");

scanf("%i", &gat);

switch (gat) //usando valor do 'gat' para escolha;

{
```

case 1: // primeira condiçao do switch;

pop_ordm (vetor,tam); // chamamos o pop_ordem

p = &esc; // logo em seguida, o ponteiro recebe o endereço da variavel esc;

*p = gat; // jogamos o valor do 'menu' escolhido pelo usuario, diretamente na variavel esc atravez do seu endereço de memoria;

break;

case 2:

pop_semi (vetor,tam); // chamamos o pop_semi;

p = &esc; // repetimos o mesmo processo de cima, para passagem de valor para a variavel esc;

*p = gat;

break;

case 3:

pop_deso(vetor,tam); //chamamos o pop_deso.

p = &esc;

*p = gat;

break;

default :

printf("\n\nOpção invalida!"); // caso nenhuma opção valida (ou seja, de 1 a 3) seja escolhida, caimos aqui, onde avisamos o usuario, limpamos a tela,

system("pause"); // e voltamos para o começo da funcao;

system("cls");

goto nd; // comando para retorno 'brusco';

}

}