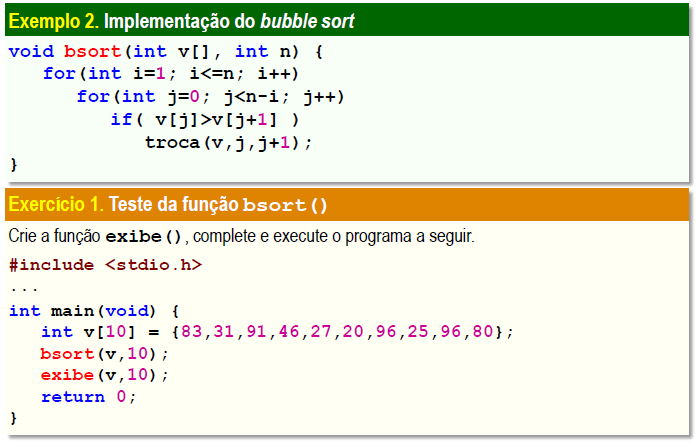
ed-07 – 23/09/2021



#include <stdio.h>

int i, j;

void troca(int v[], int i, int j) {

int x = v[i];

v[i] = v[j];

v[j] = x;

}

void bsort(int v[], int n){

for(i = 1; i <= n; i++)

for(j = 0; j < n-1; j++)

if(v[j] > v[j+1])

troca(v, j, j+1);

}

void exibe(int v[], int n){

for(i = 0; i < n; i++)

printf("%d\t", v[i]);

}

int main(void){

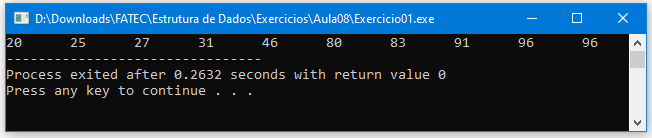
int v[10] = {83, 31, 91, 46, 27, 20, 96, 25, 96, 80};

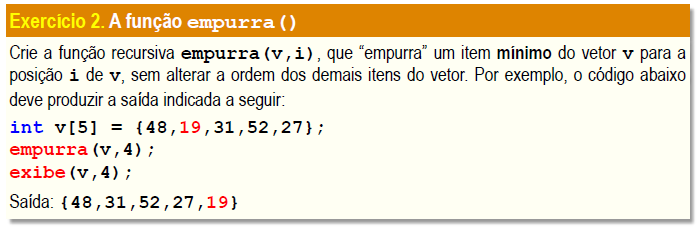
bsort(v, 10);

exibe(v, 10);

return 0;

}





#include <stdio.h>

#include "fila.h"

int n;

int menor(int v[]){

int i;

int min = v[0];

for(i=0; i<5; i++){

if(v[i] < min){

min = v[i];

}

}

return min;

}

void empurra(int v[], int i){

Fila F = fila(5);

int min = menor(v);

for(n = 0; n < 5; n++){

if(v[n] != min) enfileira(v[n], F);

}

enfileira(min, F);

for(n = 0; n < i; n++)

printf("%d\t", desenfileira(F));

destroif(F);

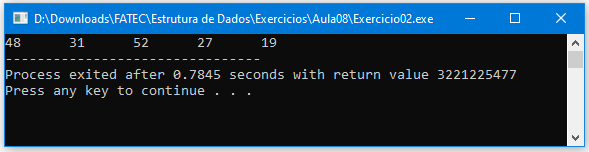
}

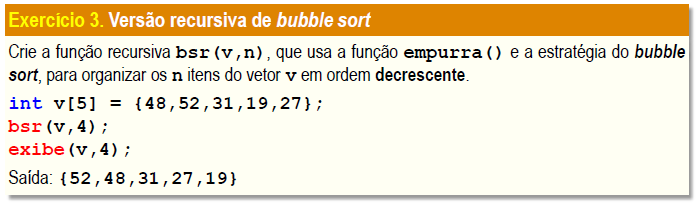
int main(void){

int v[5] = {48, 19, 31, 52, 27};

empurra(v, 5);

}





#include <stdio.h>

int i, j;

void troca (int v[], int i, int j){

int x = v[i];

v[i] = v[j];

v[j] = x;

}

void bsort(int v[], int n){

for(i = 0; i <= n; i++)

for(j = i+1; j < n; ++j)

if(v[j] > v[i])

troca(v, j, i);

}

void exibe(int v[], int n){

for(i = 0; i < n; i++)

printf("%d\t", v[i]);

}

int main (void){

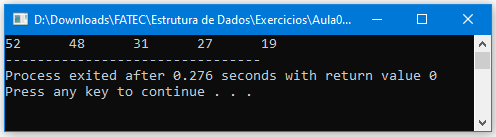
int v[5] = {48,52,31,19,27};

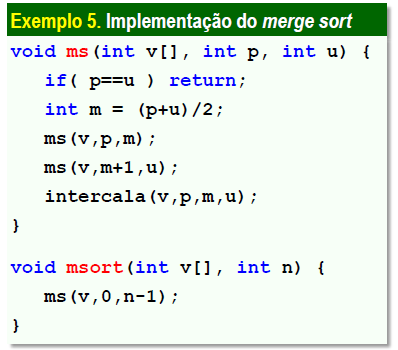
bsort(v, 5);

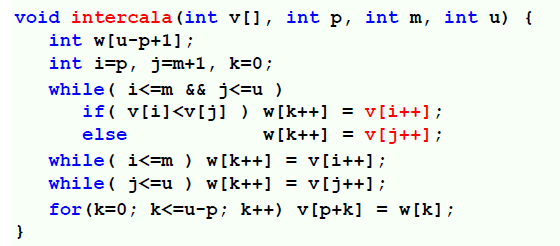
exibe(v, 5);

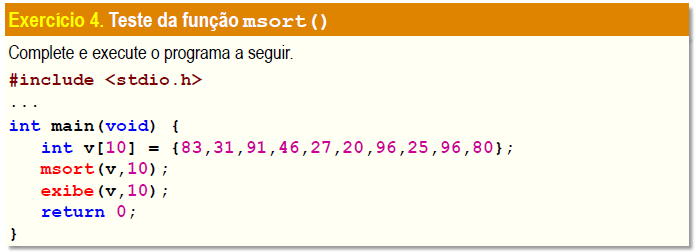
return 0;

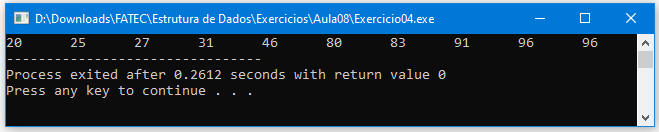
}

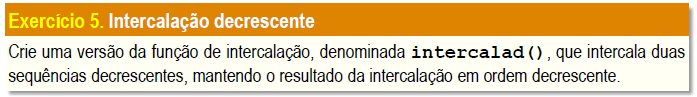












#include <stdio.h>

#include <stdbool.h>

int i;

void intercala(int v[], int p, int m, int u){

int w[u-p+1];

int i=p, j=m+1, k=0;

while(i<=m && j<=u){

if(v[i] > v[j]) w[k++] = v[i++];

else w[k++] = v[j++];

}

while(i<=m) w[k++] = v[i++];

while(j<=u) w[k++] = v[j++];

for(k=0; k<=u-p; k++) v[p+k] = w[k];

}

void ms(int v[], int p, int u){

if(p==u) return;

int m = (p+u)/2;

ms(v, p, m);

ms(v, m+1, u);

intercala(v, p, m, u);

}

void msort(int v[], int n){

ms(v, 0, n-1);

}

bool verificadorDecrescente(int v[], int sizeV, int r[], int sizeR){

bool decrescente = true;

for(i=0; i< sizeV; i++){

if(v[i] < v[i+1]) decrescente = false;

}

for(i=0; i< sizeR; i++){

if(r[i] < r[i+1]) decrescente = false;

}

return decrescente;

}

bool verificadorSequencias(int sizeV, int sizeR){

bool iguais = 0;

if(sizeV == sizeR){

iguais = 1;

}

return iguais;

}

void intercalad(int v[], int sizeV, int r[]){

int n = 0;

int w[2\*sizeV];

for(i=0; i<sizeV; i++){

w[n] = v[i];

n++;

w[n] = r[i];

n++;

}

exibe(w, 2\*sizeV);

}

void exibe(int v[], int n){

for(i = 0; i < n; i++)

printf("%d\t", v[i]);

printf("\n");

}

int main (void){

int v[7] = {121, 174, 97, 9, 17, 5, 31};

int r[7] = {70, 19, 58, 36, 65, 3, 12};

int sizeV = sizeof(v)/sizeof(v[0]);

int sizeR = sizeof(r)/sizeof(r[0]);

int ask = 0;

if(verificadorSequencias(sizeV, sizeR) != 1){

printf("Os arrays possuem tamanhos distintos!\n");

return 0;

}

if(verificadorDecrescente(v, sizeV, r, sizeR) != 1){

printf("Os arrays nao estao em ordem, gostaria de ordena-los?\n");

printf("1 - Sim\n2 - Nao\n... ");

scanf("%d", &ask);

if(ask == 2) return 0;

msort(v, sizeV);

msort(r, sizeR);

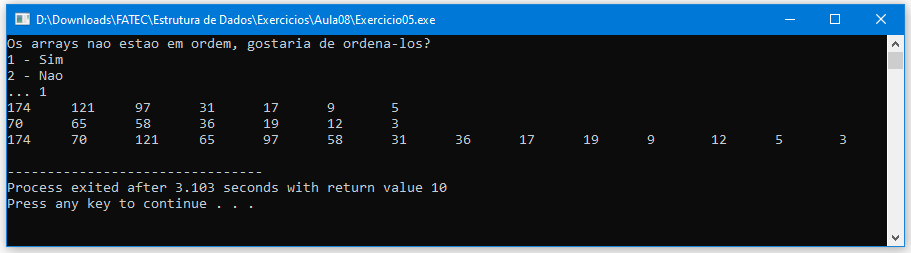
}

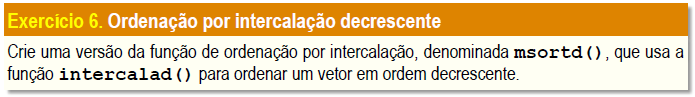
exibe(v, 7);

exibe(r, 7);

intercalad(v, sizeV, r);

}





#include <stdio.h>

#include <stdbool.h>

int i;

void intercala(int v[], int p, int m, int u){

int w[u-p+1];

int i=p, j=m+1, k=0;

while(i<=m && j<=u){

if(v[i] > v[j]) w[k++] = v[i++];

else w[k++] = v[j++];

}

while(i<=m) w[k++] = v[i++];

while(j<=u) w[k++] = v[j++];

for(k=0; k<=u-p; k++) v[p+k] = w[k];

}

void ms(int v[], int p, int u){

if(p==u) return;

int m = (p+u)/2;

ms(v, p, m);

ms(v, m+1, u);

intercala(v, p, m, u);

}

void msort(int v[], int n){

ms(v, 0, n-1);

}

bool verificadorDecrescente(int v[], int sizeV, int r[], int sizeR){

bool decrescente = true;

for(i=0; i< sizeV; i++){

if(v[i] < v[i+1]) decrescente = false;

}

for(i=0; i< sizeR; i++){

if(r[i] < r[i+1]) decrescente = false;

}

return decrescente;

}

bool verificadorSequencias(int sizeV, int sizeR){

bool iguais = 0;

if(sizeV == sizeR){

iguais = 1;

}

return iguais;

}

void intercalad(int v[], int sizeV, int r[]){

int n = 0;

int w[2\*sizeV];

for(i=0; i<sizeV; i++){

w[n] = v[i];

n++;

w[n] = r[i];

n++;

}

msortd(w, 14);

exibe(w, 2\*sizeV);

}

void msd(int v[], int p, int u){

if(p==u) return;

int m = (p+u)/2;

ms(v, p, m);

ms(v, m+1, u);

intercala(v, p, m, u);

}

void msortd(int v[], int n){

msd(v, 0, n-1);

}

void exibe(int v[], int n){

for(i = 0; i < n; i++)

printf("%d\t", v[i]);

printf("\n");

}

int main (void){

int v[7] = {121, 174, 97, 9, 17, 5, 31};

int r[7] = {70, 19, 58, 36, 65, 3, 12};

int sizeV = sizeof(v)/sizeof(v[0]);

int sizeR = sizeof(r)/sizeof(r[0]);

int ask = 0;

if(verificadorSequencias(sizeV, sizeR) != 1){

printf("Os arrays possuem tamanhos distintos!\n");

return 0;

}

if(verificadorDecrescente(v, sizeV, r, sizeR) != 1){

printf("Os arrays nao estao em ordem, gostaria de ordena-los?\n");

printf("1 - Sim\n2 - Nao\n... ");

scanf("%d", &ask);

if(ask == 2) return 0;

msort(v, sizeV);

msort(r, sizeR);

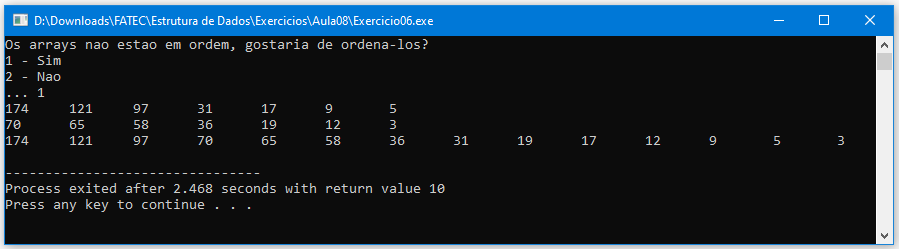
}

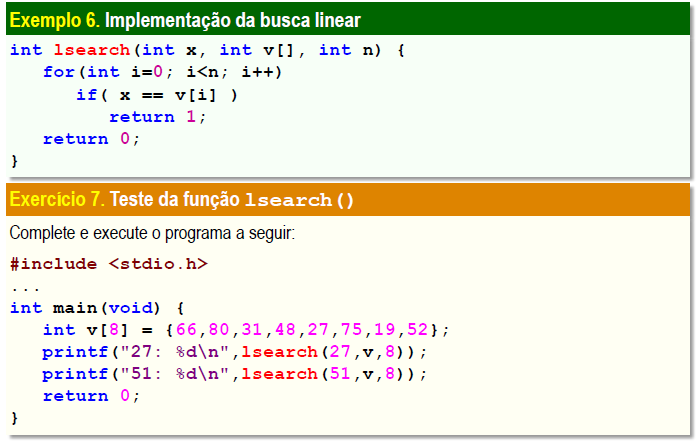
exibe(v, 7);

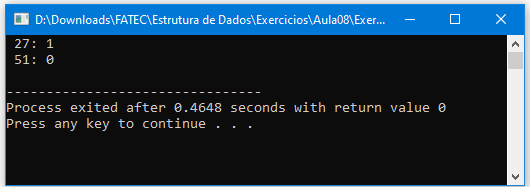
exibe(r, 7);

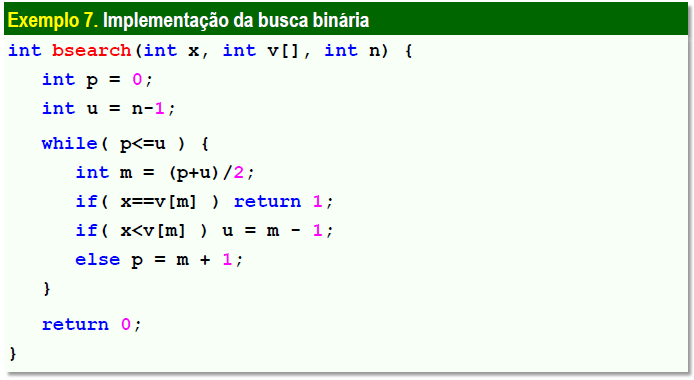
intercalad(v, sizeV, r);

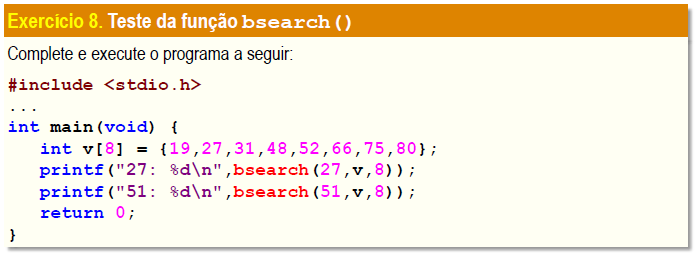
}

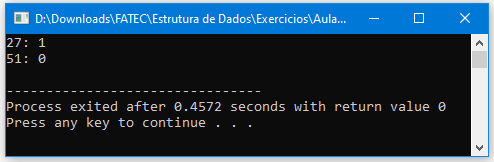


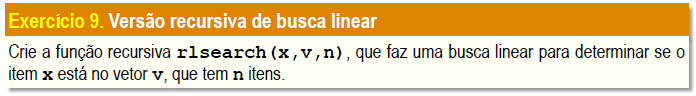












#include <stdio.h>

int i = 0;

int rlsearch(int x, int v[], int n){

if (n == 0) return 0;

if (x == v[n-1]) return 1;

return rlsearch (x, v, n-1);

}

int main(void){

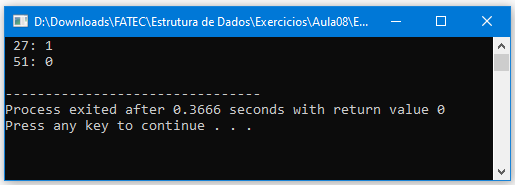
int v[8] = {66, 80, 31, 48, 27, 75, 19, 52};

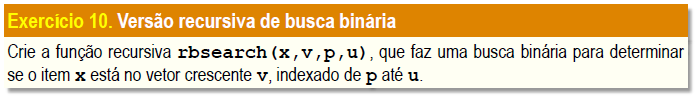
printf(" 27: %d\n", rlsearch(27, v, 8));

printf(" 51: %d\n", rlsearch(51, v, 8));

return 0;

}





#include <stdio.h>

int m = 0;

int rbsearch(int x, int v[], int p, int u){

int m;

m = (p+u)/2;

if (x == v[m]) return 1;

else if ( p>u ) return 0;

if( x < v[m]) return rbsearch(x, v, p, u-1);

else return rbsearch(x, v, p+1, u);

}

int main (void){

int v[8] = {19, 27, 31, 48, 52, 66, 75, 80};

printf("27: %d\n", rbsearch(27, v, 0, 8));

printf("51: %d\n", rbsearch(51, v, 0, 8));

return 0;

}

Texto

Descrição gerada automaticamente