**APLIKOVANÉ PRÍKLADY: zoradenie**

**Program, ktorý načíta z klávesnice prvky matice a realizuje nasledujúce operácie:**

**výpis prvkov matice pomocou indexov**

**výpis prvkov matice priamo**

**výpis prvkov v 3. riadku matice**

#include<stdio.h>

#include<conio.h>

main(void)

{

int matica[100][100];

int s,r,i,j,min;

printf ("Zadaj pocet riadkov a stlcov\n");

scanf (" %d%d", &s, &r ); // &s,&r - adresy premenných s a r

for (i=0; i<s; i++) // cyklus pre riadky

{

For (j=0; j<r; j++) // cyklus pre stĺpce

{

Printf ("\n Zadaj prvok M[ %d, %d] = ",i, j);

Scanf ("%d",&matica [i] [j]); // &matica[i][j] – adresa i j prvku poľa

}

}

Printf (" \n Vypis prvkov matice vo forme indexov \n");

for(i=0; i<s; i++) // cyklus riadkov

{

for(j=0;j<r;j++) // cyklus stĺpcov

printf("\n M[%d,%d] = %d ",i,j,matica[i][j]);

}

printf(" \n Vypis prvkov matice\n");

for(i=0; i<s; i++) // cyklus riadkov

{

printf("\n"); // každý riadok začne odriadkovaním

for(j=0; j<r; j++) // cyklus stĺpcov

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

}

printf("\n Vypis 3. riadku matice \n");

for(j=0;j<r;j++)

printf("\t%d ",matica [2] [j]); // prvky poľa sa indexujú

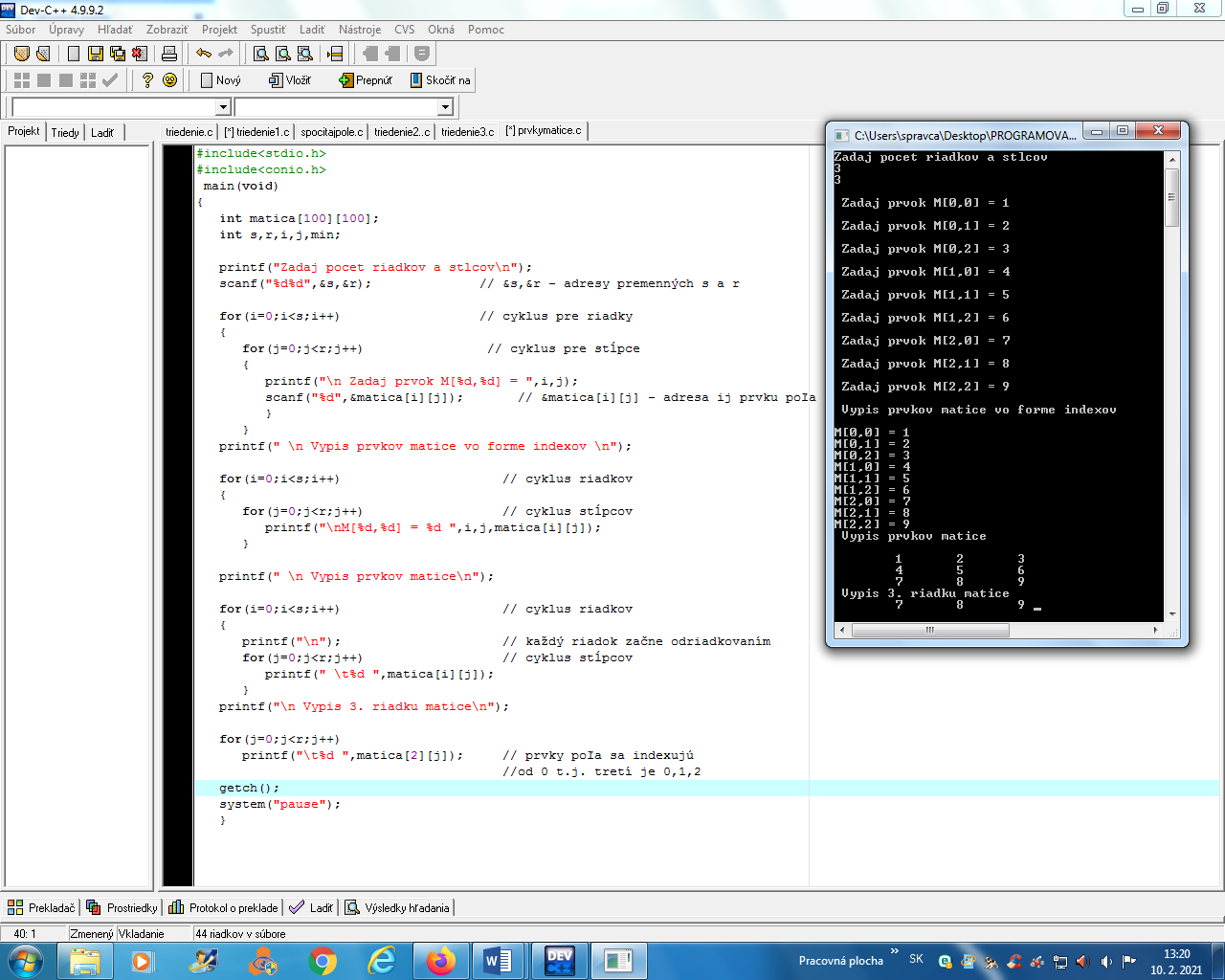
//od 0 t.j. tretí je 0,1,2

getch();

system("pause");

}

Výstup programu:



#include <stdio.h>

int main()

{

int pole[100],n,i,j,pom,pokracuj; //deklarácia poľa a premenných

printf("Zadaj velkost pola:\n");

scanf("%d", &n); //načítanie veľkosti poľa do premennej n

printf("Zadaj prvky pola[%d]:\n", n);

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

scanf("%d", &pole[i]); //načítanie prvkov poľa

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

{

pokracuj = 0;

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

{

if (pole[j] > pole[j+1]) //podmienka, či sa majú prvky vymeniť

{

pom = pole[j];pole[j] = pole[j+1];

pole[j+1] = pom;

pokracuj = 1;

}

}

if(pokracuj == 0)

break;

}

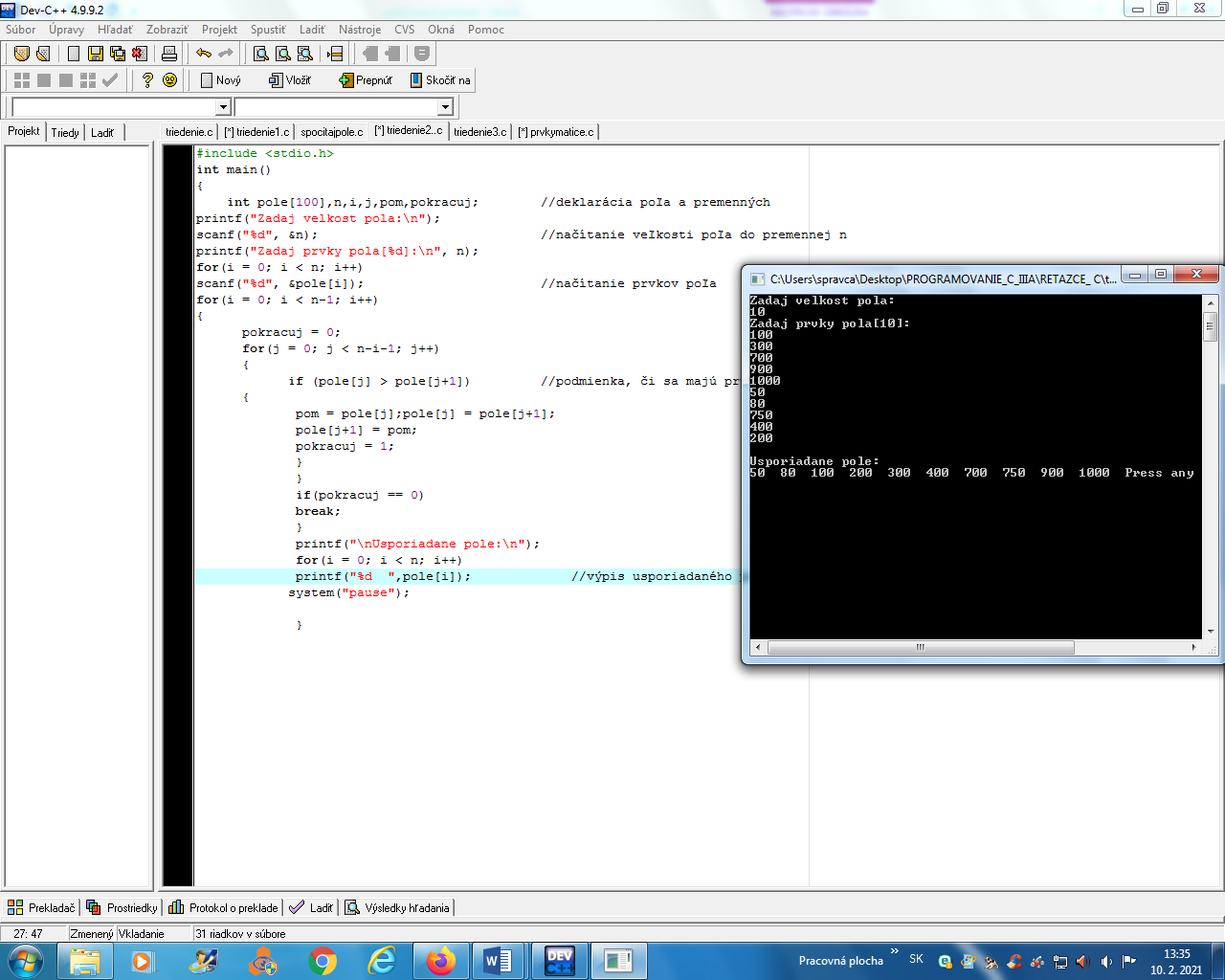
printf("\nUsporiadane pole:\n");

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

printf("%d ",pole[i]); //výpis usporiadaného poľa

system("pause");

}



**TRIEDENIE**

Už od mala nás učili dávať si veci do poriadku, a teda s istým triedením sme sa už všetci streli. Objekty sa triedia všade tam, kde sa budú neskôr vyhľadávať a vyberať, napríklad v telefónnych zoznamoch, obsahoch kníh, knižniciach, slovníkoch, skladoch. Triedenie teda považujeme za základnú a dôležitú činnosť pri spracovaní údajov. Využíva sa hlavne na to, aby sa poukázalo na veľkú rozmanitosť algoritmov. Všetky tieto algoritmy majú rovnaký cieľ,no každý jeden je optimálny v istom zmysle, pre určitý typ triedenia.

Spôsoby delenia triediacich algoritmov„ Z pohľadu vlastného spôsobu zoraďovania dát existuje veľa rôznych metód, ktoré sa líšia:

•zložitosťou algoritmu•nárokom na pamäť (počtom pomocných premenných)

•rýchlosťou (počtom porovnaní)“ (PŠENČÍKOVÁ, 2009)Obvykle sa metódy triedenia delia do dvoch kategórií:

•triedenie polí -vnútorné triedenie

•triedenie súborov –vonkajšie triedenie

Priame metódy možno rozdeliť na tri základné skupiny:

•triedenie vkladaním

•triedenie výberom

•triedenie výmenou(Wirth, 1975)

Priame metódy majú svoje výhody, sú jednoduché, krátke a triedia na mieste. Na druhú stranu je tu veľká nevýhoda a tou je výpočtová zložitosť. Z toho vyplýva, že priame metódy triedenia sú použiteľné v prípadoch, kedy sa nejdená o veľké množstvo triedených dát.

Príklad triedenia poľa celých čísel 6, 2, 4, 8, 8, 3, 1, 3, 9, 5 vzostupne (od najmenšieho po najväčšie)

Bubblesort

Bublinkové triedenie (Bubblesort) nám už znázvu hovorí v čom spočíva jeho základná myšlienka. Ide o to, aby sme nechali prebublať väčšie prvky v poli rovnako ako stúpajú bublinky v nápoji. V triediacom algoritme sa opakovane prechádza celé pole prvkov, kde sa vzájomne porovnávajú susedné dvojice. Ak daná dvojica nie je správne zoradená, tak sa prvky vymenia. V prípade, že prvky sú usporiadané správne, tak ostanú na svojich pozíciách.

Algoritmus bublinkového triedenia

Vstup: Pole P[n]

1.pokračuj ⇓1

2.Pokiaľ je pokračuj = 1:

3.pokračuj ⇓0

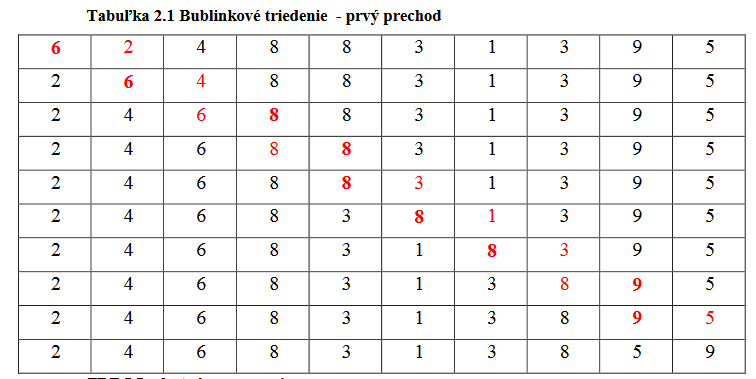
4.Pre i = 0,...,n-2:

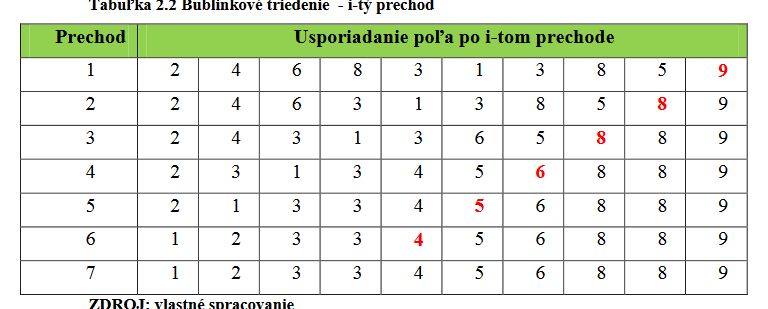
5.Ak je P[i] > P[i+1]:

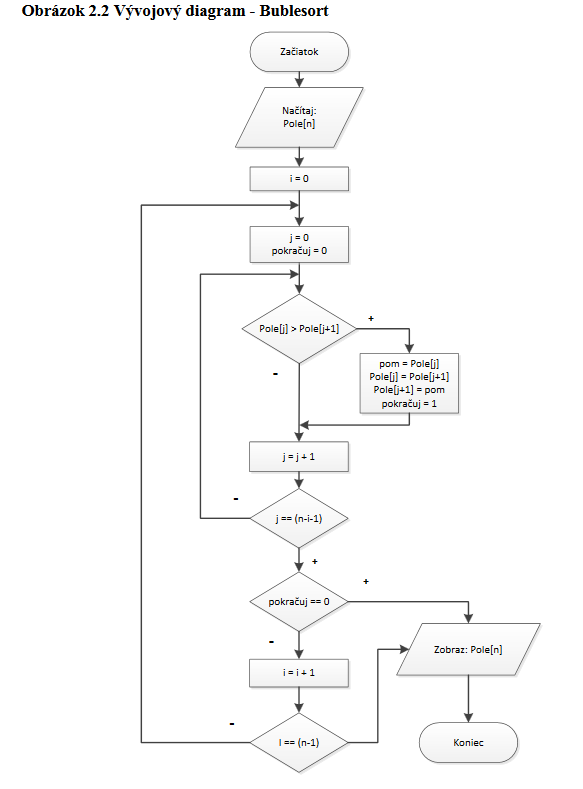
6.Prehoď prvky P[i] a P[i+1]

7.pokračuj ⇓1

Výstup: Usporiadané pole P







**Príklad zdrojového kódu pre bublinkové triedenie**

#include <stdio.h>

#define MAX 10

main()

{ int i,j, pocet;

int a[MAX],pomoc;

printf("\n Zadaj pocet prvkov postupnosti: ");

scanf("%d",&pocet);

if(pocet>MAX)

printf("\n To je prilis mnoho, maximum je %d\n",MAX);

else /\* pocet prvkov bol programom akceptovany \*/

{

for(i=0;i<pocet;i++) { /\* citanie prvkov postupnosti \*/

printf("a[%d] : ",i+1);

scanf("%d",&a[i]);

}

/\* algoritmus jednoducheho bublinkoveho triedenia : \*/

for(i=pocet-1;i>0;i--)

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

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

{ pomoc=a[j]; a[j]=a[j+1]; a[j+1]=pomoc;

}

for(i=0;i<pocet;i++) { /\* vypis usporiadanej postupnosti \*/

printf("\n vypisujem postupne: [%d] : ",i+1);

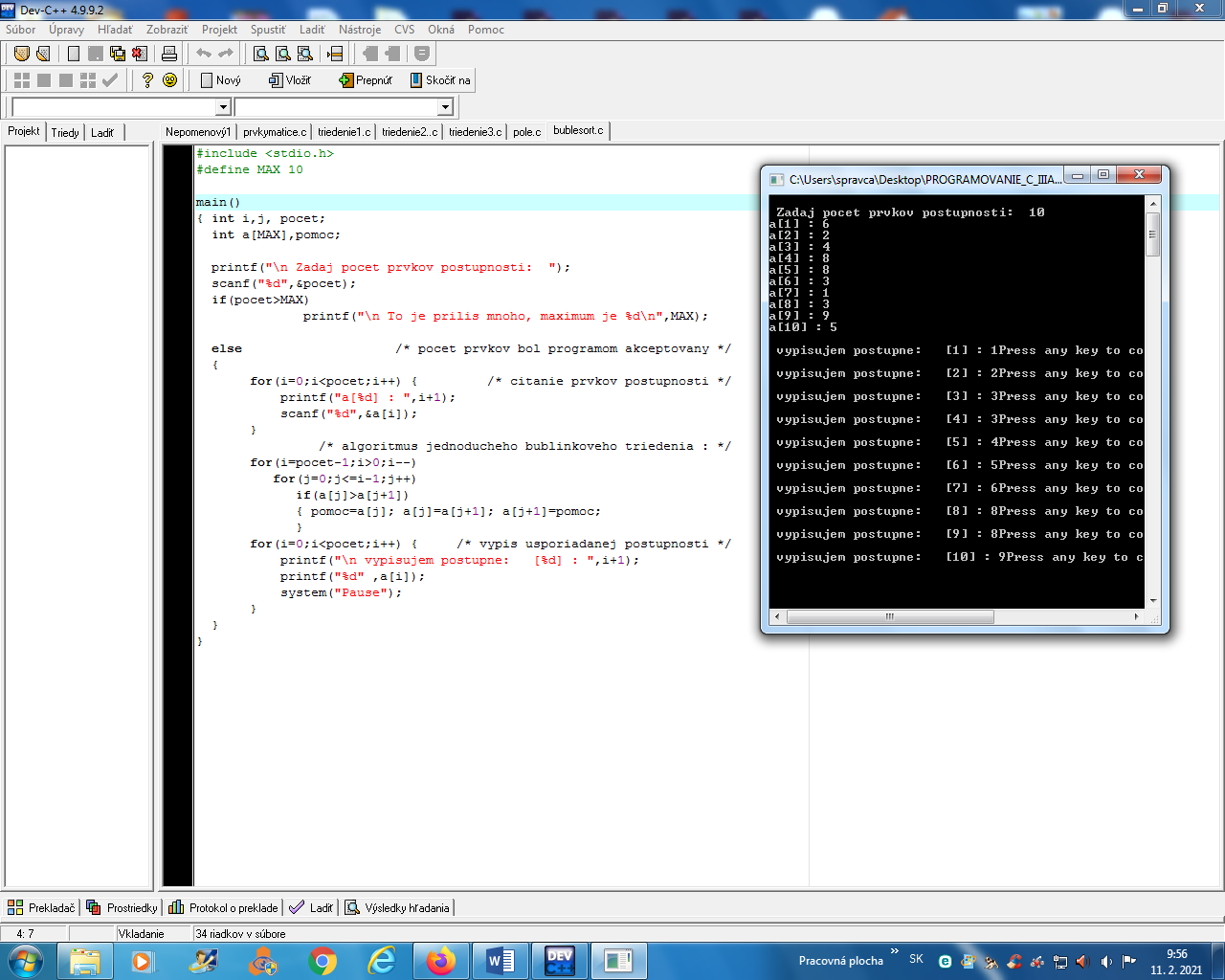
printf("%d" ,a[i]);

system("Pause");

}

}

}



Shakesort

Na bublinkové triedenie priamo nadväzuje triedenie pretriasaním (Shakesort), ktorého názov pochádza z prirovnania k barmanovi ako trasie šejkrom. Zatrasením šejkrom sa bublinky v nápoji dostávajú raz na jednu a raz na druhú stranu. Rovnakým spôsobom sa prechádza poľom prvkov. Najskôr zľava doprava dostaneme najväčší prvok a následne dostaneme najmenší prvok sprava doľava. Pole prvkov sa teda bude usporadúvať aj od konca aj od začiatku. Posledná utriedená časť bude stred poľa.

Insertsort

Ďalšou priamou metódou je triedenie priamym vkladaním (Insertsort). V tejto metóde sa prvky poľa rozdelia na dve skupiny(postupnosti): zdrojovú a cieľovú. Prvky sa vyberajú zo zdrojovej a ukladajú sa do cieľovej skupiny. Najskôr sa vyberie druhý prvok v poradí zo zdrojovej postupnosti a vloží sa na príslušné miesto v cieľovej postupnosti. Následne sa rovnakým spôsobom vyberie a vloží tretí, štvrtý až posledný prvok a pole je usporiadané. Daný prvok sa na správne miesto v cieľovej postupnosti dostane tak, že ho porovnáme so susedným prvkom vľavo. Ak je susedný prvok väčší, tak ho posunieme vpravo o jedno miesto. Proces hľadania správneho miesta sa skončí v dvoch prípadoch:1.Susedný prvok je menší ako prvok, ktorému hľadáme miesto.2.Dostaneme sa na koniec cieľovej postupnosti.

Selectsort

Základným krokom triedenia priamym výberom je vyberanie prvku s najmenšou hodnotou a jeho umiestnenie na začiatok poľa. Pole si môžeme rozdeliť na dve časti. Jedna je na začiatku poľa a vkladáme do nej nájdené najmenšie prvky. Druhú časť tvoria ostatné prvky poľa, ktoré ešte nie sú usporiadané. Po každom nájdení a uložení najmenšieho prvku sa neusporiadaná časť poľa zmenší o1 a usporiadaná časť sa zväčšío1. Týmto sa docieli, že už usporiadaný prvok sa ďalej neporovnáva. Najmenší prvok z neusporiadanej časti sa vloží vždy na koniec usporiadanej časti.

Čerpané zo zdroja:

<file:///C:/Users/spravca/Downloads/BC_UHLIARIK.pdf>

**Zložitejší príklad triedenia s ponukou výberu zoradenia**

#include<stdio.h>

#include<conio.h>

void main(void)

{

int a[100];

int i,x,j,pom,n,k;

printf("\n Zadaj velkost pola: ");

scanf("%d",&n);

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

{

printf("\n Zadaj a[%d]= ",i);

scanf("%d",&a[i]);

}

printf("\n Chcete usporiadat prvky pola zostupne alebo vzostupne ( z/v ): \n");

x=getch(); // nacitaj jeden znak odpovede

if(x=='v')

{

for (i=0;i<n-1;i++) // n cyklov potrebnych pre

//"prebublanie" kazdeho prvku

{

for (j=0;j<n-i-1; j++) // v kazdom dalsom cykle sa

// zotrieďuje o 1 mensi pocet

{

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

{

pom=a[j]; // vymena dvoch prvkov cez pom premennu

a[j]=a[j+1];

a[j+1]=pom;

} } }

printf("\n Usporiadanie od najmensieho po najvacsi");

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

printf ("\na[%d]= %d",i,a[i]);

}

if(x=='z')

{

for (i=0;i<n-1;i++) // n cyklov potrebnych pre

// "prebublanie" kazdeho prvku

{

for (j=0;j<n-i-1;j++) // v kazdom dalsom cykle sa zotrieduje o

// 1 mensi počet

{

if (a[j]<a[j+1])

{

pom=a[j]; // vymena dvoch prvkov cez pom premennu

a[j]=a[j+1];

a[j+1]=pom;

} } }

printf("\n Usporiadanie od najvacsieho po najmensi");

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

printf ("\na[%d]= %d",i,a[i]);

}

getch();

}

