**Laboratorul 6 – Pointeri**

/// 3.2. Folosind numai pointeri şi expresii cu pointeri se vor scrie funcţii de sortare a unui vector cu elemente reale (precum şi programul care le testează).

#include <stdio.h>

#include <stdlib.h>

float \*citeste(int \*pn, char nume){

int i;

printf("Numar de elemente pentru %c: ", nume); /// t <=> &t[0]

scanf("%d", pn); /// \*t <=> t[0]

/// t+i <=> &t[i]

float \*a = calloc(\*pn, sizeof(float)); /// \*(t+i) <=> t[i]

if(a==NULL){

printf("Memorie insuficienta");

exit(1);

}

for(i=0; i<\*pn; i++){

printf("%c[%d]=", nume, i);

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

}

return a;

}

void bubble(float \*a, int n){

int i, sortat=0;

float aux;

while(!sortat){

sortat=1;

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

if(a[i-1]>a[i]){

aux=a[i-1];

a[i-1]=a[i]; /// \*(a+i-1)=\*(a+i)

a[i]=aux;

sortat=0;

}

n--;

}

}

void afiseaza(float a[], int n){

int i;

for(i=0; i<n; i++) printf("%.0f ", a[i]);

}

int main()

{

int n;

float \*a;

a=citeste(&n, 'a');

bubble(a, n);

printf("Dupa sortare:\n");

afiseaza(a, n);

free(a);

return 0;

}

/// 3.3. Folosind numai pointeri şi expresii cu pointeri se va scrie o funcţie de interclasare a doi vectori, care conţin elemente de tip real ordonate crescător (precum şi programul care testează această funcţie).

#include <stdio.h>

#include <stdlib.h>

float \*citeste(int \*pn, char nume){

int i;

printf("Numar de elemente pentru %c: ", nume);

scanf("%d", pn);

float \*a = calloc(\*pn, sizeof(float));

if(a==NULL){

printf("Memorie insuficienta");

exit(1);

}

for(i=0; i<\*pn; i++){

printf("%c[%d]=", nume, i);

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

}

return a;

}

void afisare(float a[], int n){

int i;

for(i=0; i<n; i++) printf("%.0f ", a[i]);

}

float \*intercl(float a[], int na, float b[], int nb, int \*pnc){

int i=0, j=0, k=0;

\*pnc=na+nb;

float \*c = calloc(\*pnc, sizeof(float));

if(c==NULL){

printf("Memorie insuficienta");

exit(1);

}

while(i<na && j<nb)

if(a[i]<=b[j]) c[k++]=a[i++];

else c[k++]=b[j++];

while(i<na) c[k++]=a[i++];

while(j<nb) c[k++]=b[j++];

return c;

}

int main()

{

float \*a, \*b, \*c;

int na, nb, nc;

a=citeste(&na, 'a');

b=citeste(&nb, 'b');

c=intercl(a, na, b, nb, &nc);

printf("Rezultatul interclasarii: ");

afisare(c, nc);

free(a);

free(b);

return 0;

}

/// 3.4. Folosind numai pointeri şi expresii cu pointeri se vor scrie funcţii de citire, afişare şi înmulţire a două matrice (precum şi programul care le testează).

#include <stdio.h>

#include <stdlib.h>

#define N 10

void citeste(float a[][N], int \*pn, int \*pm, char nume){

printf("Nr. de linii pentru %c: ", nume); /// a <=> &a[0]

scanf("%d", pn); /// a[i] <=> &a[i][0] <=> \*(a+i)

/// a[i][j] <=> \*(a[i]+j) <=> \*(\*(a+i)+j)

printf("Nr. de coloane pentru %c: ", nume); /// &a[i][j] <=> a[i] + j <=> \*(a+i)+j

scanf("%d", pm);

int i, j;

for(i=0; i<\*pn; i++)

for(j=0; j<\*pm; j++){

printf("%c[%d, %d]=", nume, i, j);

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

}

}

void produs(const float a[][N], const float b[][N], float c[][N], int n, int m, int \*pnc, int \*pmc){

int i, j, k;

\*pnc=n;

\*pmc=m;

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

for(j=0; j<m; j++){

c[i][j]=0;

for(k=0; k<n; k++) c[i][j]+=a[i][k]\*b[k][j];

}

}

void afiseaza(float a[][N], int n, int m){

int i, j;

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

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

printf("%.1f ", a[i][j]);

printf("\n");

}

}

int main()

{

float A[N][N], B[N][N], C[N][N];

int na, nb, ma, mb, nc, mc;

citeste(A, &na, &ma, 'A');

citeste(B, &nb, &mb, 'B');

if(na!=mb){

printf("Nu se poate calcula produsul");

return 0;

}

printf("AxB:\n");

produs(A, B, C, na, ma, &nc, &mc);

afiseaza(C, nc, mc);

return 0;

}

/// 3.5. Să se scrie o funcţie care sortează în ordine crescătoare n şiruri de caractere.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char \*\*citeste(int \*pn){ /// \*\* pentru ca este vector de caractere (cum ar fi c[][])

int i;

char tmp[100];

printf("Numar de siruri: ");

scanf("%d%\*c", pn);

char \*\*s=calloc(\*pn, sizeof(char\*));

if(s==NULL){

printf("Memorie insuficienta");

exit(1);

}

for(i=0; i<\*pn; i++){

fgets(tmp, 100, stdin);

s[i]=calloc(strlen(tmp)+1, sizeof(char));

if(s[i]==NULL){

printf("Memorie insuficienta");

exit(1);

}

strcpy(s[i], tmp);

}

return s;

}

void bubble(char\*\* s, int n){

int i, sortat=0;

char \*aux;

while(!sortat){

sortat=1;

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

if(strcmp(s[i], s[i-1])>0){

aux=s[i-1];

s[i-1]=s[i];

s[i]=aux;

sortat=0;

}

n--;

}

}

void afisare(int n, char \*\*s){

int i;

for(i=0; i<n; i++) printf("%s", s[i]);

}

int main()

{

int n;

char \*\*s;

s=citeste(&n);

bubble(s, n);

afisare(n, s);

free(s);

return 0;

}

/// 3.6. Să se scrie o funcţie care caută prima apariţie a unui şir de caractere în alt şir de caractere, și returnează poziţia pe care apare subşirul în şir, sau -1 dacă nu a fost găsit. Folosind această funcţie, să se determine toate poziţiile pe care apare un şir de caractere într-un altul.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N 100

int pozSubsir(const char \*s, const char \*subsir){

char \*p=strstr(s,subsir);

if(p!=NULL) return p-s;

else return -1;

}

int main()

{

int poz, nrAparitii=0, i=0;

char s[N], subsir[N];

printf("Introduceti sirul: ");

gets(s);

printf("Introduceti subsirul cautat: ");

gets(subsir);

while((poz=pozSubsir(s+i, subsir))>=0){

nrAparitii++;

printf("Subsirul apare pe pozitia: %d\n", poz+i);

i+=poz+1;

}

if(nrAparitii==0) printf("Subsirul nu apare");

return 0;

}

/// 3.7. Să se scrie o funcţie care determină rădăcina unei funcţii f(x), în intervalul [a,b], ştiind că admite o singură rădăcină în acest interval. Funcţia f va fi transmisă ca parametru efectiv.

#include <stdio.h>

#include <stdlib.h>

int main()

{

return 0;

}

/// 3.8. Să se scrie o funcție pentru calculul derivatei unui polinom P(x) de grad n, într-un punct dat x=x0. Gradul și coeficienții polinomului precum și punctul x0 se vor trimite ca argumente la apelul funcției.

#include <stdio.h>

#include <stdlib.h>

float \*citestePol(int \*pg, char nume){

int i;

printf("Gradul polinomului %c: ", nume);

scanf("%d", pg);

float \*a=calloc(\*pg+1, sizeof(float));

if(a==NULL){

printf("Memorie insuficienta");

exit(1);

}

printf("Coeficientii polinomului %c: \n", nume);

for(i=\*pg; i>=0; i--){

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

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

}

return a;

}

float valPolinom(float x, int n, float a[]){

int i;

float v;

for(i=n; i>=1; i--) v=v\*x+i\*a[i];

return v;

}

int main()

{

int grad;

float \*a, x0;

a=citestePol(&grad, 'P');

printf("x0 = ");

scanf("%f", &x0);

printf("P(%.2f) = %.3f\n", x0, valPolinom(x0, grad, a));

free(a);

return 0;

}