Temă pentru acasă. Pointeri către funcţii

I. Considerăm următorul program în care este implementată o metodă foarte simplă de aproximare a valorii maxime a unei funcţii:

```
#include<iostream>
#include<math.h>
using namespace std;

double f(double x){
    return (x-1)*(5-x);
}
double g(double t){
    return exp(1-t*t);
}
double valMax(double (*pf)(double), double a, double b){
    const int N=1000;
    const double h=(b-a)/N;
    double vmax=(*pf)(a);
    for( a+=h; a<=b; a+=h){
        double val=(*pf)(a);
        if(vmax<val) vmax=val;
    }
    return vmax;
}
int main(void) {
    cout<<valMax(f,1,5)<<endl;
    cout<<valMax(g,-10,10)<<endl;
    return 0;
}
```

Maximul este determinat de subrutina

`double valMax(double (*pf)(double), double a, double b)`

care primeşte funcţia de analizat prin intermediul parametrului ***pf*** de tip pointer către funcţii de tip ***double***(***double***).

Exersaţi utilizarea acestui tip de pointer completând programul de mai sus conform cerinţelor următoare:

1. Declaraţi cu ***typedef*** tipul ***Functie*** care să fie sinonim cu "funcţie care primeşte un ***double*** şi returnează un ***double***", rescrieţi apoi antetul funcţiei ***valMax*** sub forma

   `double valMax(Functie *pf, double a, double b)`

2. In limbajul C pentru a afla adresa unei funcţii nu este nevoie de operatorul adresă, numele unei funcţii fiind o constană care desemnează adresa funcţiei. În C++ această facilitate a fost întărită: pentru a desemna funcţia referită de un pointer nu mai este nevoie de operatorul ţintă, adresa funcţiei fiind sinonimă cu numele ei. In exemplul nostru, instrucţiunea

   ```
   double vmax=(*pf)(a);
   ```

   este echivalentă cu

   ```
   double vmax=pf(a);
   ```

   Modificaţi programul pentru a utiliza această facilitate specifică C++.

3. Definiţi funcţia

   ```
   double metodaInjumatatirii(Functie *pf, double a, double b)
   ```

   care rezolvă prin metoda înjumătăţirii intervalului ecuaţia $f(x) = 0$, unde $f : [\,a, b\,] \to \mathbb{R}$ este o funcţie continuă oarecare cu $f(a)f(b) \leq 0$. La apelarea metodei, f este ţinta pointerului ***pf***.

4. Implementaţi funcţia

   ```
   double formulaTrapezelor(Functie *pf, double a, double b)
   ```

   care calculează aproximativ integrala Riemann a unei funcţii $f : [\,a, b\,] \to \mathbb{R}$ prin *formula trapezelor sumată*:

   $$\int_a^b f(x)dx \simeq h\Big(\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{N-1} f(x_i)\Big),$$

   cu $h = (b-a)/N$ şi $x_i = a+ih$, pentru $i = 0, 1, 2, \ldots, N$. Formula se stabileşte prin împărţirea intervalului $[\,a, b\,]$ în N subintervale $[\,x_i, x_{i+1}\,]$ şi aproximarea integralei pe un subinterval cu aria trapezului determinat de punctele $(x_i, 0)$, $(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$ şi $(x_{i+1}, 0)$, adică

   $$\int_{x_i}^{x_{i+1}} f(x)dx \simeq \frac{(f(x_i) + f(x_{i+1}))h}{2}.$$

   Funcţia f este transmisă în ***formulaTrapezelor*** prin pointerul ***pf***, iar N este fixat, $N = 1000$.

5. Scrieţi o subrutină care estimează *variaţia totală* a unei funcţii $f : [\,a, b\,] \to \mathbb{R}$ oarecare. Variaţia totală $V_a^b(f)$ este definită astfel:

   $$V_a^b(f) = \sup_{\Delta} \sum_{i=0}^{N} |f(x_{i+1}) - f(x_i)|,$$

   unde supremumul este luat după toate diviziunile

   $$\Delta : a = x_0 < x_1 < \ldots < x_N = b$$

   ale intervalului $[\,a, b\,]$.

II. Fixăm constanta $N = 1000$ şi considerăm pe mulţimea $\mathcal{M} = \{1, 2, \ldots, N\}$ o relaţie $\rho \subseteq \mathcal{M} \times \mathcal{M}$ dată de funcţia sa de comparaţie

`bool compRo(int a, int b)`

care întoarce ***true*** dacă $a \in \mathcal{M}$, $b \in \mathcal{M}$ şi $a\rho b$, altfel întoarce ***false***.

De exemplu, funcţia de comparaţie corespunzătoare relaţiei ρ definită de $a\rho b$ dacă $a \mid b$, este următoarea:

```
bool compRoDiv(int a, int b){
    if(a<1 || b<1 || a>N || b>N) return false;
    return b%a==0;
}
```

Se cere să se decidă prin verificarea tuturor cazurilor posibile (*metoda forţei brute*) dacă o relaţie ρ are proprietăţile unei ordini totale pe $\mathcal{M}$. Parcurgeţi etapele:

1. Definiţi cu ***typedef*** tipul ***Comparatie*** care să fie sinonim cu "funcţie cu două argumente de tip ***int*** şi cu rezultat de tip ***bool***";

2. Definiţi funcţiile

   i) `bool esteReflexiva(Comparatie *ro)`

   ii) `bool esteAntisimetrica(Comparatie *ro)`

   iii) `bool esteTranzitiva(Comparatie *ro)`

   iv) `bool esteTotala(Comparatie *ro)`

   v) `bool esteOrdineTotala(Comparatie *ro)`

   care decid proprietăţile relaţiei ρ. La apelare, pointerul ***ro*** are ca ţintă funcţia de comparaţie ***compRo*** corespunzătoare. Amintim că o relaţie ρ este *reflexivă* dacă $a\rho a$ pentru orice $a \in \mathcal{M}$; este *antisimetrică* dacă din $a\rho b$ şi $b\rho a$ rezultă $a = b$; este *tranzitivă* dacă $a\rho b$ şi $b\rho c$ implică $a\rho c$ şi este *totală* dacă oricare două elemente a şi b din $\mathcal{M}$ sunt comparabile (i.e. are loc măcar una dintre situaţiile $a\rho b$ sau $b\rho a$). O relaţie ρ care are toate aceste patru proprietăţi este o *relaţie de ordine totală*.

3. Determinaţi prin program proprietăţile relaţiilor $\rho_i \subseteq \mathcal{M} \times \mathcal{M}$ definite astfel

   i) $a\rho_1 b$ dacă $a + b \mid a^2 + b^2$;

   ii) $a\rho_2 b$ dacă $2a \leq 3b$;

   iii) $a\rho_3 b$ dacă $a(a^2 - ab + 1) \leq b(b^2 - ab + 1)$.

III. In contextul de mai sus, definiţi funcţia

`bool esteRelatieDeEchivalenta(Comparatie *ro)`

care decide dacă ρ este o *relaţie de echivalenţă* (adică este *reflexivă*, *simetrică* şi *tranzitivă*). Testaţi următoarele relaţii

i) $a\rho_1 b$ dacă $a + b \mid ab$;

ii) $a\rho_2 b$ dacă suma cifrelor lui a este egală cu suma cifrelor lui b;

iii) $a\rho_3 b$ dacă prima cifră a lui a este egală cu prima cifră a lui b.

IV. In programul următor sortăm (aranjăm în ordine) un tablou de numere întregi prin metoda "minimul pe primul loc": în prima etapă aflăm minimul din tot tabloul şi îl mutăm pe primul loc, repetăm apoi procedeul aflând de fiecare dată minimul din partea nesortată şi mutându-l pe primul loc al acesteia:

```
#include<iostream>
#include<time.h>
using namespace std;
typedef bool Comparatie(int,int);
bool compNumAscendenta(int a,int b) {
    return a<=b;
}
bool compNumDescendenta(int a,int b) {
    return a>=b;
}
void sorteaza(int tab[], int dim, Comparatie *pComp ) {
    for(int i=0;i<dim-1;i++){
        int imin=i, vmin=tab[imin];
        for(int j=i+1;j<dim;j++){
            int valj=tab[j];
            if(pComp(vmin,valj))continue;
            imin=j;
            vmin=valj;
        }
        //mutam minimul pe primul loc
        if(imin!=i) {
            tab[imin]=tab[i];
            tab[i]=vmin;
        }
    }
}
int genereazaDate(int tab[], int dim) {
    cout<<"\nDate:\n"<<endl;
    srand ( (unsigned)time(NULL) );
    for (int i=0; i < dim; i++)
    cout<<(tab[i]=rand()%100)<<endl;
    return dim;
}
int main(void) {
    const int dim=30;
    int tab[dim];
    //--- generam numerele pe care le vom sorta
    genereazaDate(tab, dim);
    char optiune;
    do{
    cout<<"\nAlegeti ordinea de sortare:\n"<<endl;
    cout<<"<a> - numeric ascendenta"<<endl;
    cout<<"<d> - numeric descendenta"<<endl;
    cout<<"<s> - stop program"<<endl;
```

```
        cout<<"< >\b\b";
        cin>>optiune;
        switch(optiune){
        case 'A': case 'a':
            cout<<"\nSortam ascendent\n"<<endl;
            sorteaza(tab, dim, compNumAscendenta);
            for (int i=0; i < dim; i++) cout<<tab[i]<<endl;
            break;
        case 'D': case 'd':
            cout<<"\nSortam descendent\n"<<endl;
            sorteaza(tab, dim, compNumDescendenta);
            for (int i=0; i < dim; i++) cout<<tab[i]<<endl;
            break;
        }
        }while(optiune!='S'&& optiune!='s');
        cout<<"GATA"<<endl;
        return 0;
    }
```

In subrutina care realizează sortarea

`void sorteaza(int tab[], int dim, Comparatie *pComp )`

relaţia de ordine este dată de pointerul `pComp` care ţinteşte către una dintre cele două funcţii de comparaţie

`bool compNumAscendenta(int a,int b)`

şi

`bool compNumDescendenta(int a,int b)`

care stabilesc *ordinea numerică ascendentă* sau *ordinea numerică descendentă*, în acord de opţiunea utilizatorului. Atenţie, funcţiile de comparaţie trebuie să returneze ***true*** în cazul de egalitate.

1. Schimbaţi metoda de sortare în "minimul pe primul şi maximul pe ultimul loc", aflând în fiecare etapă minimul şi maximul din partea nesortată a tabloului şi mutând minimul pe primul loc şi maximul pe ultimul loc al acesteia.

2. Implementaţi sortarea prin "metoda bulelor" (*bubble sort*): parcurgem tot tabloul şi schimbăm între ele elementele vecine care nu sunt în ordine, repetăm această parcurgere până când nu mai apare nici o schimbare.

3. Stabiliţi care dintre cele trei metode de sortare implementate este mai eficientă, determinând prin calcul şi verificând prin program numărul total de comparaţii efectuat la sortarea unui tablou;

4. Definiţi funcţia de comparaţie

   `bool compLexDirecta(int a, int b)`

   care precizează *ordinea lexicografică directă* dintre $\boldsymbol{a}$ şi $\boldsymbol{b}$: comparăm pe rând prima cifră a lui $\boldsymbol{a}$ cu prima cifră a lui $\boldsymbol{b}$, dacă sunt egale comparăm a doua cifră a lui $\boldsymbol{a}$ cu a doua a lui $\boldsymbol{b}$ şi aşa mai departe până când găsim cifre distincte sau până epuizăm unul dintre numere. Este mai mic numărul epuizat sau, dacă am găsit cifre distincte, este mai mic numărul care o deţine pe cea mai mică dintre cele două cifre distincte.

Următoarele numere sunt în ordine lexicografică directă:

```
10162
10663
1779
20
20
2125
21253
5390
```

5. Definiţi funcţia de comparaţie

   `bool compLexInversa(int a, int b)`

   care precizează *ordinea lexicografică inversă* dintre ***a*** şi ***b***, care se stabileşte după acelaşi principiu ca în cazul precedent, cu deosebirea că acum parcurgem numerele de la drepta la stânga: comparăm mai întâi ultima cifră a lui ***a*** cu ultima cifră a lui ***b***, ş. a. m. d.

   Următoarele numere sunt în ordine lexicografică inversă:

```
 7210
14310
 7121
   43
  143
 3904
11164
 2164
```

   Completaţi meniul afişat de funcţia ***main*** astfel încât utilizatorul să poată alege şi sortarea în ordine lexicografică directă sau inversă.

V. Sortaţi prin metoda bulelor un tablou de puncte din plan, fiecare punct fiind o structură formată din coordonatele ***x*** şi ***y*** de tip ***double***.

Oferiţi utilizatorului posibilitatea să aleagă una dintre următoarele 4 relaţii de ordine: "mai la nord-vest", "mai la nord-est", "mai la sud-vest" sau "mai la sud-est", unde, de exemplu, (x_1, y_1) se afla "mai la nord-vest" decât (x_2, y_2) dacă $y_1 > y_2$ sau dacă $y_1 = y_2$ şi $x_1 \leq x_2$.