

Bevezetés a programozásba I

10. gyakorlat

C++: alprogramok deklarációja és paraméterátadása

© 2011.11.22. Giachetta Roberto
groberto@inf.elte.hu
http://people.inf.elte.hu/groberto

Alprogramok deklarációja

Deklaráció és definíció szétválasztása

- C++-ban is szétválaszthatjuk az alprogram deklarációját a definíciójától
 - Az alprogram deklarációjában nem kell megírunk a változók neveit, csak a típusait:
`<típus> <név>(<paraméterek típusa>);`
 - A definíció megegyezik a alprogram függvényleírással:
`<típus> <név>(<paraméterek>) {<utasítások>}`
 - például megvalósítható, hogy az alprogramokat a főprogram előtt deklaráljuk, és a törzsét a főprogram után írjuk, így rögtön láthatjuk, mit tartalmaz a főprogram, pl.:

```
void fv();  
int main(){ /* itt használhatjuk fv-t */  
void fv(){ /* utasítások */ }
```

Alprogramok deklarációja

Példa

Feladat: Számoljuk ki függvény segítségével egy szám n-edik hatványát.

- használjunk szorzást a függvényben, amelynek paraméterei a szám, illetve a kitevő lesznek
- a függvénytörzset helyezzük a főprogram alá

Megoldás:

```
/* hatványozás  
bemenet: egy egész (a), egy természetes szám  
          (n)  
kimenet: a-nak n-edik hatványa */  
int pow(int, int); // deklaráció
```

Alprogramok deklarációja

Példa

Megoldás:

```
// főprogram  
int main(){  
int szam, n;  
cout << "A szám: ";  
cin >> szam;  
cout << "A kitevő: ";  
cin >> n;  
cout << "Eredmény: " << pow(szam, n) << endl;  
return 0;  
}
```

Alprogramok deklarációja

Példa

Megoldás:

```
int pow(int a, int n){ // definíció (törzs)  
int x = 1;  
for (int i = 0; i < n; i++)  
x *= a; // hatványozás szorzással  
return x;  
}
```

Álnevek

Használata

- Lehetőségünk van olyan változókat létrehozni, amelyek egy másik változó értékét használják fel, azaz pontosan ugyanazt az értéket tárolják, mint a másik változó, ezek az úgynevezett *álnevek* (*alias*-k)
- Az álneveket mindig egy már létező változóra állítjuk rá, és onnan a ponttól bármelyik értékének módosítása a másik értékét is módosítja
- Az álnevek típusának meg kell egyeznie az eredeti változó típusával, a neve pedig tetszőleges lehet
- A C++-ban az álneveket az `&` operátorral jelöljük meg:
`<típus> <változó 1>;
<típus> &<változó 2> = <változó 1>;
// a változó 2 lesz a változó 1 álneve`

Álnevek	
Egyszerű változókra	
<ul style="list-style-type: none"> Egy változóra persze több álnevet is ráállíthatunk Pl.: <pre>int main() { char x = 'a'; char &y = x; // ráállítunk egy y álnevet az x-re cout << y << endl; // ez kiírja az 'a' értéket x = 'b'; cout << y << endl; // ez kiírja a 'b' értéket y = 'c'; cout << x << endl; // ez kiírja a 'c' értéket return 0; }</pre> 	
PPKE ITK, Bevezetés a programozásba I	10:7

Álnevek	
Tömbökre	
<ul style="list-style-type: none"> Lehetőségünk van tömbökre is ráállítani álneveket, de ekkor a * operátort használunk: <pre><típus> <változó 1>[<méret>]; <típus> *<változó 2> = <változó 1>;</pre> Ténylegesen a következő történik: <ul style="list-style-type: none"> mivel minden változót a memóriában tárolunk, rendelkezik egy memóriabeli címmel, amit lekérdezhetünk az & és * operátorokkal létrehozott változók megkapják a már létező változók memóriabeli címét, és ezért pont ugyanarra a memóriaterületre fognak mutatni értékmódosításkor ugyanarra a memóriaterületre fognak írni, ezért ugyanazt az értéket fogják ott felülírni 	
PPKE ITK, Bevezetés a programozásba I	10:8

Alprogramok paraméterátadása	
Alprogramok kommunikációja	
<ul style="list-style-type: none"> A C++ programok függvényekből állnak, amelyek tetszőleges utasítássorozatot, műveletet végrehajthatnak, ezeket a függvényeket más függvényekben, alprogramokban használhatjuk meghívással A függvények kommunikálhatnak visszatérési értékkel (minden függvénynek csak egy lehet), globális változókon (használatuk hibákat rejt magában) és paramétereken keresztül Eddig paraméterátadásnál az aktuális paraméter értéke átkerült a formális paraméterbe, és azt az egész függvényben használhatjuk Azonban a paraméterátadásnál nem csak az alprogramnak adhatunk át értéket, hanem az alprogram is adhat át értéket az őt meghívó programrésznek 	
PPKE ITK, Bevezetés a programozásba I	10:9

Alprogramok paraméterátadása	
A paraméterátadás iránya	
<ul style="list-style-type: none"> Ennek megfelelően a <i>paraméterátadás iránya</i> szerint a következő paramétertípusokat különböztetjük meg: <ul style="list-style-type: none"> <i>bemenő paraméter</i>: a hívás pillanatában átadódik az az aktuális paraméter értéke a formálisnak, és onnantól az adatok módosítása az alprogramban nincs hatással a főprogramra (ezt használtuk eddig) <i>kimenő paraméter</i>: az értéket az alprogram állítja be, majd a hívás végeztével (azaz az alprogram lefutásának végén) a formális értékek visszairódnak a főprogramba az aktuális paraméterbe <i>be- és kimenő paraméter</i>: az értékek a főprogramból bekerülnek az alprogramba, amely azokat átírhatja, majd a hívás végeztével a módosított értékek visszakerülnek a főprogramba 	
PPKE ITK, Bevezetés a programozásba I	10:10

Alprogramok paraméterátadása	
A paraméterátadás módja	
<ul style="list-style-type: none"> A három irány megvalósításához különböző módszereket kell alkalmaznunk: <i>Érték szerinti paraméterátadás</i>: <ul style="list-style-type: none"> az aktuális paraméter értékét átadjuk a formális paraméternek a hívás pillanatában a formális paraméter ezután azzal az értékkel fog dolgozni az alprogramban hiába módosítjuk az aktuális, vagy a formális paramétert, az nincs hatással a másikra pl.: <pre>void fv(int x); ... int y; fv(y); // y érték szerint adódik át</pre> 	
PPKE ITK, Bevezetés a programozásba I	10:11

Alprogramok paraméterátadása	
Példa	
<p><i>Feladat</i>: Számítsuk ki függvény segítségével két egész szám legnagyobb közös osztóját.</p> <ul style="list-style-type: none"> a függvénynek érték szerinti paraméterátadással adjuk át a két egész számot, és a visszatérési értéke legyen az osztó használjuk az euklédieszi algoritmust a függvényben <p><i>Megoldás</i>:</p> <pre>... /* legnagyobb közös osztó meghatározása bemenet: két természetes szám (a, b) kimenet: a és b legnagyobb közös osztója */ int loko(int a, int b);</pre>	
PPKE ITK, Bevezetés a programozásba I	10:12

Alprogramok paraméterátadása

Példa

Megoldás:

```
int main(){ // főprogram
    int x = 10, y = 8; int z = lnko(x,y);
    // itt x = 10, y = 8, z = 2
    cout << "LNKO: " << z << endl;
    return 0;
}

int lnko(int a, int b){ // itt a = 10, b = 8
    while (a != b){ // euklédieszi algoritmus
        if (a > b) a -= b; else b -= a;
    }
    return a;
} // itt a = 2, b = 2
```

PPKE ITK, Bevezetés a programozásba I

10:13

Alprogramok paraméterátadása

A paraméterátadás módja

- *Cím szerinti paraméterátadás:*
 - az aktuális paraméter memóriacímét adja át a formális paraméternek, a formális paraméter csak a változó memóriacímét tárolja, és az megváltoztathatatlan, viszont a memóriacímen található érték megváltoztatható
 - az alprogramban új értéket adhatunk a változónak, és az az érték visszakérül a főprogramba
 - a C++-ban ezt meg lehet valósítani álnevek használatával úgy, hogy formális paraméterként álneveket definiálunk, így az egyszerre módosul az aktuális paraméterrel
 - ezzel lehetőségünk van be- és kimenő paraméterátadást megvalósítani

PPKE ITK, Bevezetés a programozásba I

10:14

Alprogramok paraméterátadása

A paraméterátadás módja

- Tekintsük az előző példát cím szerinti paraméterátadást használva:

```
int lnko(int &a, int &b){
    // itt a = 10, b = 8
    ...
    return a;
} // itt a = 2, b = 2

int main(){
    int x = 10; int y = 8; int z = lnko(x,y);
    // itt x = 2!, y = 2!, z = 2 lesz
    cout << "LNKO: " << z << endl;
    return 0;
}
```

PPKE ITK, Bevezetés a programozásba I

10:15

Alprogramok paraméterátadása

A paraméterátadás módja

- Cím szerinti átadással lehetőségünk van megspórolni a visszatérési értéket, hiszen az érték módosul, pl:

```
void lnko(int &a, int &b){ // itt a = 10, b = 8
    while (a != b)
        if (a > b) a -= b; else b -= a;
} // itt a = 2, b = 2

int main(){
    int x = 10; int y = 8;
    lnko(x,y); // itt x = 2, y = 8 lesz
    cout << "LNKO: " << x << endl;
    return 0;
}
```

PPKE ITK, Bevezetés a programozásba I

10:16

Alprogramok paraméterátadása

Példa

Feladat: Olvassuk be 10 valós számot, és határozzuk meg a számok szinuszáinak átlagát.

- a beolvasást és az átlagszámítást helyezzük függvénybe
- ahhoz, hogy a beolvasást követően a tömb értékei visszakérüljenek a főprogramba, cím szerinti paraméterátadást kell használnunk
- a másik függvényben használhatunk érték szerinti átadást, és az eredményt visszaadhatjuk visszatérési értékben
- a szinuszhoz használjuk a `sin` függvényt (a `math.h` fájlból)

PPKE ITK, Bevezetés a programozásba I

10:17

Alprogramok paraméterátadása

Példa

Megoldás:

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

/* beolvasás
   bemenet: 10 valós szám az inputon
   kimenet: 10 elemű valós tömb (szamok) */
void beolvas(double* szamok);

/* a szinuszok átlaga
   bemenet: 10 elemű valós tömb (szamok)
   kimenet: a tömbelemek szinuszáinak átlaga */
double sin_atlag(double szamok[10]);
```

PPKE ITK, Bevezetés a programozásba I

10:18

Alprogramok paraméterátadása

Példa

Megoldás:

```
/* főprogram
   bemenet: 10 szám az inputon
   kimenet: a számok szinuszának átlaga */
int main(){
    double t[10];
    beolvas(t); // beolvasás cím szerinti átadással
    // t értékei módosultak
    cout << "Szinuszok átlaga: " << sin_atlag(t)
         << endl;
    // t értékei nem módosultak
    return 0;
}
```

PPKE ITK, Bevezetés a programozásba I

10:19

Alprogramok paraméterátadása

Példa

Megoldás:

```
void beolvas(double* szamok){
    // cím szerinti paraméterátadás
    cout << "Beolvasás: " << endl;
    for (int i = 0; i < 10; i++){
        cout << i+1 << ". szám: "; cin >> szamok[i];
    }
} // a szamok értékei módosultan visszaadódnak
double sin_atlag(double szamok[10]){
    double s = 0; // összegzés
    for (int i = 0; i < 10; i++){
        s += sin(szamok[i]);
    }
    return (s / 10);
}
```

PPKE ITK, Bevezetés a programozásba I

10:20

Alprogramok paraméterátadása

A paraméterátadás módja

- *Eredmény szerinti paraméterátadás*: megegyezik az érték szerinti átadással, de a formális paraméter értéke a hívás után visszakerül az aktuális paraméterbe (tehát ellentétben a cím szerinti átadással a függvény futása közben nem módosul az eredeti értéke, csak a végén)
- *Név szerinti paraméterátadás*: egy kifejezés pontosan átadódik, és a formális paraméter hívásakor mindig kiértékelődik a benne használt változók értékének függvényében
- A C++ ez utóbbi kettőt nem támogatja, ezért a következő lehetőségeink vannak:
 - csak bemenő paraméter érték szerinti átadással
 - be- és kimenő paraméter cím szerinti átadással

PPKE ITK, Bevezetés a programozásba I

10:21

Alprogramok paraméterátadása

A főprogram paraméterei

- Nem csak az alprogramok, hanem a főprogram (**main** függvény) is kaphat paraméterezést
 - a főprogramnak átadott aktuális paraméterek a parancssorban megadott, fájlnev utáni szavak lesznek
 - bármennyi, bármilyen paramétert megadhatunk neki, amelyet a főprogrammal ki tudunk értékelteni
 - pl.: `./a.out ./szamok.txt 75 13 joprogi`
- A főprogram formális paraméterei:
 - **int argc**: ez mondja meg, hogy hány paraméterrel indították a programot
 - **char* argv[]**: ez a tömb tartalmazza a paramétereket egymás után

PPKE ITK, Bevezetés a programozásba I

10:22

Alprogramok paraméterátadása

A főprogram paraméterei

- A főprogramot ennek megfelelően a következőképpen írhatjuk:

```
int main(int argc, char* argv[]){
    // itt használhatjuk argc-t, argv-t
}
```
- Az **argv[<index>]** elemet lekérdezve megtudhatjuk a valahányadik argumentumot (paramétert)
 - a tömb **argc** hosszú, tehát a tömb `0...argc-1` elemeit kérdezhetjük le
 - **argc** értéke minden esetben legalább egy, ugyanis az első argumentum maga a program neve (elérési úttal), utána következnek a tényleges paraméterek
 - pl. ha a program futtatása `./a.out`, akkor `argv[0] = "home/groberto/a.out"` lehet

PPKE ITK, Bevezetés a programozásba I

10:23

Alprogramok paraméterátadása

A főprogram paraméterei

- Tehát az **argc**-vel ellenőrizhetjük, megadtak-e paramétert a program indításakor a név mellett
- Így lehetőségünk van például fájlneveket megadnunk argumentumként, így nem kell a program futása közben bekérnünk őket:

```
int main(int argc, char* argv[]){
    ifstream f;
    if (argc > 1)
        // ha van más paraméter a fájlneven kívül
        f.open(argv[1]); // akkor nyissuk azt meg
    else // különben kérjük be a fájlnevet
        ...
}
```

PPKE ITK, Bevezetés a programozásba I

10:24

Alprogramok paraméterátadása

Példa

Feladat: Olvassunk ki egy sort egy fájlból, amelyet paraméterként adtunk meg. Ha nem jó paramétert adunk meg, vagy egyáltalán nem adunk paramétert, akkor kérjük be billentyűzetről. Továbbá írjuk ki, hogy milyen néven fut a program.

- ha az első megnyitás nem sikerült, kérjük be billentyűzetről a fájlnevet, amíg sikertelen a megnyitás

Megoldás:

```
...
/* főprogram
   bemenet: egy fájlnev argumentumként
   kimenet: a fájl első sora */
int main(int argc, char* argv[]){
```

Alprogramok paraméterátadása

Példa

Megoldás:

```
cout << argv[0] << " program fut...";
ifstream f; string nev;
if (argc > 1) { // van paraméter
  cout << "Megnyitás: " << argv[1] << endl;
  f.open(argv[1]);
} else { // nincs paraméter, bekérjük
  cout << "File neve: ";
  cin >> nev;
  f.open(nev.c_str());
}
while (f.fail()){
  // ha nem sikerült az első megnyitás
  cout << "Hibás fájlnev!" << endl;
```

Alprogramok paraméterátadása

Példa

Megoldás:

```
f.clear();
cout << "File neve: ";
cin >> nev;
f.open(nev.c_str());
} // addig bekérjük, amíg nem sikerül megnyitni

string sor;
getline(f, sor);
cout << "Az első sor: " << sor << endl;
return 0;
}
```

Alprogramok

Feladatok

Az alábbi feladatokban függvényt kell megvalósítani. Ha másképp nem rendelkezik a feladat, a main függvényben a megvalósított függvényt kell tesztelni az érdekes esetekre.

VI. Függvények:

8. Valósítsd meg az `int hanyor(ifstream &f)` függvényt, ami egy paraméterül kapott fájlban levő maradék sorok számát adja vissza.
17. Valósítsd meg a skalárszorzat műveletet függvényként.
18. Valósítsd meg a mátrixszorzás műveletet függvényként.

Alprogramok

Feladatok

VI. Függvények:

19. b) (*) Valósíts meg függvényt, ami megkap egy mátrixot, és kiszámítja az inverzét. A függvény adjon vissza egy állapot is, ami jelzi, ha hiba van: a mátrix nem négyzetes, vagy összefüggőek a vektorok.