Programozás alapjai
3. gyakorlat
A C nyelv szelekciós és ciklus utasításai, egyszerű vezérlési szerkezetek gyakorlása

Szelekciós utasítás (kétirányú elágazás esetén):

```
if (kifejezés) {
        utasítások
}
else {
        utasítások
}
```

1. Adjunk meg egy számot, és döntsük el róla hogy páros vagy páratlan. Feltételes operátorral:

```
#include <stdio.h>
int main() {
        int a;
        printf("\nKérek egy egész számot: ");
        scanf("%d", &a);
        printf( (a % 2 == 0) ? " páros" : " páratlan" );
        return 0;
}
```

Szelekciós utasítással:

```
#include <stdio.h>
int main() {
        int a;
        printf("\nKérek egy egész számot: ");
        scanf("%d", &a);
        if (a%2) printf(" páratlan\n");
        else printf(" páros\n" );
        return 0;
}
```

Switch szerkezettel:

```
#include <stdio.h>
int main() {
        int a;
        printf("Adj meg egy számot: ");
        scanf("%d", &a);
        switch(a%2) {
                case 0: printf("Páros.\n"); break;
                case 1: printf("Páratlan.\n"); break;
        }
        return 0;
}
```

A switch után megadott kifejezés csak egész vagy egészre konvertálódó típusú lehet C-ben.

2. Egy megadott szám abszolútértékének meghatározása (szelekciós utasítással).

```
#include <stdio.h>
int main() {
        int a;
        printf("\nKérek egy számot: ");
        scanf("%d", &a);
        if (a<0)  a *= -1;
        /* else ;   → hamis ágon üres utasítás, elmaradhat */
        printf("\nAbszolút értéke: %d\n", a);
        return 0;
}
```

Oldja meg a feladatot a feltételes operátor használatával is!

Ciklus utasítások:

Előltesztelő ciklus:

```
while (ciklus feltétel) {
        utasítások
}
```

Hátultesztelő ciklus:

```
do {
        utasítások
} while (ciklus feltétel);
```

3. Számok összegzése 1-től N-ig (elöl- és hátultesztelő ciklussal). Használják a léptető operátort és az összevont értékadás operátorát (+=).

```
int main() {
    int N, i, sum=0;
    printf("\nKérek egy számot: ");
    scanf("%d", &N);
    i=1;
    while (i<=N) {
        sum += i;
        i++;
     }
    printf("\nÖsszeg: %d\n", sum);
    return 0;
}
```

```
int main() {
    int N, sum=0;
    printf("\nKérek egy számot: ");
    scanf("%d", &N);
    do {
        sum += N;
        N--;
    } while (N>0);
    printf("\nÖsszeg: %d\n", sum);
    return 0;
}
```

4. Egy adott szám osztóinak megszámolása elöltesztelő ciklussal.

```
int main() {
    int szam, oszto, db=0;
    printf("\nKérek egy számot: ");
    scanf("%d", &szam);
    oszto=1;
    while (oszto<=szam) {
        if (szam%oszto == 0) db++;
        oszto++;
     }
    printf("\nOsztók száma: %d\n", db);
    return 0;
}
```

```
int main() {
    int szam, oszto, db=0;
    printf("\nKérek egy számot: ");
    scanf("%d", &szam);

    for (oszto=1; oszto<=szam; oszto++) {
        if (szam%oszto == 0)
            db++;
    }
    printf("\nOsztók száma: %d\n", db);
    return 0;
}
```

Otthon gyakorolni: ciklusok átírása (elöltesztelő ↔ hátultesztelő, while ↔ for)

Ellenőrzött adatbeolvasás

5. Adott a>0 és b>=0 számokra határozzuk meg a^b hatványértéket.

```
#include <stdio.h>
int main() {
        int a, b, i;
        long int eredmeny = 1;
        printf("\nKérem a hatvány alapot és a kitevőt vesszővel elválasztva: ");
        scanf("%d, %d", &a, &b);
        for(i=1; i<=b; i++) eredmeny *= a;
        printf("\n%d %d. hatványa: %ld\n", a, b, eredmeny);
        return 0;
}
```

Hibás eredményt kapunk, ha a két számot rossz formátumban adjuk meg (nem vesszővel elválasztva), vagy ha nem egész értékeket adunk meg. Ezekben az esetekben a *scanf* függvény vizsgálatával kiszűrhető a hibás működés. Ugyanis a *scanf* függvény által visszaadott érték azt tartalmazza, hogy hány értéket dolgozott fel a függvény az input sorból.

```
if (scanf("%d,%d", &a, &b) != 2) {
        printf("Hibás adatok!\n");
        return 1;                               //sikertelen programfutás jelzése
}
else {
        for(i=1; i<=b; i++)   eredmeny *= a;
}
```

Ha hibás adatbevitel esetén újbóli adatbeolvasást szeretnénk megvalósítani, ciklust kell használni. Azonban ügyelni kell arra, hogy a *scanf* függvény számára az adatokat az operációs rendszer az <Enter> billentyű lenyomásakor egy ideiglenes tároló területre (bufferba) tölti. Innen csak annyi karaktert dolgoz fel a függvény, amennyit értelmezni képes, a többi bennmarad a bufferban. Ezt kihasználva lehet vizsgálni a buffer tartalmát: ha ez nem ‘\n’, akkor nem megfelelő adatokat adott meg a felhasználó. DE a következő *scanf* függvényhívás először a bufferban maradt karaktereket próbálja értelmezni!

```
char c;
int ok;
do {
        ok = 1;
        printf("\nKérem a hatvány alapot és a kitevőt vesszővel elválasztva: ");
        if (scanf("%d,%d", &a, &b) != 2) {
                printf("Hibás adatok!\n");
                while ( (c = getchar()) != '\n');       //input buffer ürítése
                ok = 0;
        }
} while (!ok);
```

Az input adatok (a és b) értékére korlátozás bevezetése (többirányú elágazás):

```
int ok;
char c;
do {
        printf("\nKérem a hatvány alapot és a kitevőt vesszővel elválasztva: ");
        if (scanf("%d, %d", &a, &b) != 2)  {
                printf("Hibás adatok!\n");
                while ( (c = getchar()) != '\n');        //input buffer ürítése
                ok = 0;
        }
        else if(a<=0 || b<0) {
                printf("Pozitív számot adj meg!\n");
                while ( (c = getchar()) != '\n');        //input buffer ürítése
                ok = 0;
        }
} while (!ok);
```

Megjegyzések:

1. A C szabványban nincs definiálva eljárás az input buffer kiürítésére, de egyes Windows verziókban implementálták ezt: `fflush(stdin).`
2. Karakter beolvasásakor az input bufferben lévő white space (nem látható) karakterek is értelmezhetők, azaz a *scanf("%c", &c);* utasítás kiolvassa a következő karaktert (még akkor is ha az white space). Ellenben a *scanf(" %c", &c);* utasítás átugorja a white space karaktereket (ha vannak) és az ezek után következő első karaktert dolgozza fel az input bufferből. Próbálja ki ezt:

```
char c;
do {
        printf("\nAdj meg egy karaktert: ");
        scanf("%c", &c);                /* jó megoldás: scanf(" %c", &c); */
        printf("%c", c);
} while (c!='z');
```

6. Írjunk kalkulátor programot, amely képes a 4 alapművelet végrehajtására. Ne engedélyezze a 0-val való osztást! Legyen a program folyamatos működésű. Amíg a felhasználó ki nem lép a programból, fogadja az inputot és számol. Ügyeljen a karakter beolvasásra!

```
PROGRAM: folyamatos kalkulátor
DO
        INPUT: a op b
        CASE op
                '+' : OUTPUT : a+b
                '-'  : OUTPUT : a-b
                '*' : OUTPUT : a*b
                '/' : OUTPUT : a/b
        INPUT: valasz
        IF valasz = 'i' THEN tovabb ← 1
        ELSE tovabb ← 0
WHILE tovabb = 1
PROGRAM vége
```

Házi feladat:

1. Írjon C programot, amely két megadott számról eldönti, hogy melyik a nagyobb (szelekciós utasítás). Vizsgálja az egyenlőségüket is! Megoldható-e a feladat switch szerkezettel? Válaszát indokolja!
2. Készítsen C programot, amely kiszámítja két pozitív szám számtani és mértani közepét. Egész és lebegőpontos számokkal is teszteljék a kódot.
3. Írjon C programot egy beolvasott N érték faktoriálisának kiszámítására. Figyelem! 8! = 40320, ami már nem fér el egy *int* típusú változóban. Korlátozza N értékét 0 és 10 közé, és válasszon megfelelő adattípust a faktoriális tárolásához!
4. Írjon C programot a másodfokú egyenlet valós megoldásainak kiszámítására.