

## Programozás alapjai

### 4. gyakorlat

#### Összetett logikai feltételek, vezérlési szerkezetek egymásba ágyazása, számlálás, összegzés

**1. feladat:** Eldönteni egy megadott évszámról, hogy szökőév-e. Szökőév minden negyedik év, de a századik nem. Ugyanakkor minden négyszázadik év szökőév. Például: 2000, 2004 szökőévek, de 1900 nem az.

```
if ( (year % 4 == 0) && (year % 100 != 0) || (year % 400 == 0) )
    printf("Szökőév \n");
else
    printf("Nem szökőév \n");
```

Ellenőrzött módon olvasson be egy időintervallumot (2 évszámot) és számolja meg, hány szökőév volt az adott időszakban.

#### // intervallum ellenőrzött beolvasása

```
do {
    ok = 1;                                // az alsó határ >= 1000 legyen
    printf("Add meg az intervallum alsó határát: ");
    if (scanf("%d", &lower) != 1 || lower < 1000) {
        printf("Hibás input\n");
        ok = 0;
        while ((ch=getchar()) != '\n') ;    // input buffer ürítése
    }
} while (!ok);
```

```
do {
    ok = 1;                                // felsőhatár > alsóhatár és felsőhatár <= 2019
    printf("Add meg az intervallum felső határát: ");
    if (scanf("%d", &upper) != 1 || upper < lower || upper > 2019) {
        printf("Hibás input\n");
        ok = 0;
        while ((ch=getchar()) != '\n') ;    // input buffer ürítése
    }
} while (!ok);
```

**2. feladat:** Karaktervizsgálat. Eldönteni egy megadott karakterről, hogy benne van-e az abc-ben, illetve magánhangzó-e.

```
if( (ch >= 'a' && ch <= 'Z') || (ch >= 'A' && ch <= 'z') ) {
    if( ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u' || ch=='A' || ch=='E' ||
        ch=='I' || ch=='O' || ch=='U' )
        printf("%c magánhangzó \n", ch);
    else
        printf("%c mássalhangzó \n", ch);
}
else
    printf("%c nincs az angol ABC-ben \n", ch);
```

Egy adott karakter (Q) lenyomásáig olvasson be a billentyűzetről karaktereket. Számolja meg, hány magánhangzó volt közöttük.

```
do {
    ch = getchar();
    ...
} while ( ch != 'q' && ch != 'Q' );
```

**3. feladat:** Eldönteni egy megadott számról, hogy prímszám-e. Megfigyelni a *continue* használatát *for* és *while* ciklus esetén. Melyik algoritmus a gyorsabb?

a) Osztók megszámlálása (oszthatóság vizsgálat else ág nélkül).

```
int main() {
    int szam, i=2, db=0;
    printf("Kérek egy számot: "); scanf("%d", &szam);
    for (i=2; i<=szam/2; i++)
        if (szam%i==0) db += 1;           //db=db+1;
    if (db) printf("A megadott szám nem prím.");
    else printf("A megadott szám prím.");
    return 0;
}
```

b) Eldöntés tétel kódolása (while ciklus continue utasítással).

```
int main() {
    int szam, i=2, talalt=0;
    printf("Kérek egy számot: "); scanf("%d", &szam);
    while (i<=szam/2 && !talalt) {
        if (szam%i) { i++; continue; }           //Utasítások sorrendje fontos!
        else talalt=1;
    }
    if (talalt) printf("A megadott szám nem prím.");
    else printf("A megadott szám prím.");
    return 0;
}
```

c) Ciklus megszakítással.

```
int main() {
    int szam, i=2, talalt=0;
    printf("Kérek egy számot: "); scanf("%d", &szam);
    for (i=2; i<=szam/2; i++) {
        if (szam%i) continue;
        else { talalt=1; break; }
    }
    if (talalt) printf("A megadott szám nem prím.");
    else printf("A megadott szám prím.");
    return 0;
}
```

#### 4. Példa többirányú elágazásra

Írjon telefonhasználati díjak számlázására alkalmas programot. Összesítse N db hívás után a telefonhasználati díjakat hívás típusonként (1-külföldi, 2-hálózaton kívüli, 3-hálózaton belüli). A hívási időt percben adja meg a felhasználó. Díjszabás: külföldi hívás 100 Ft/perc, hálózaton kívüli hívás 60 Ft/perc, hálózaton belüli hívás 40 Ft/perc.

```
int main () {
    int N, i, tipus;
    double ido, sum_kf, sum_hk, sum_hb;
    sum_kf = sum_hk = sum_hb = 0;
    printf("Hány adat lesz? "); scanf("%d", &N);
    i = 1;
    while (i<=N) {
        printf("%d. hívás adatai: \n", i);
        printf("Típus [1-külföldi, 2-halozaton kivuli, 3-halozaton beluli]: ");
        scanf("%d", &tipus);
        printf("Beszelgetes ideje (percben): "); scanf("%lf", &ido);

        /*Többirányú elágazás: egymásba ágyazott if utasítások */
        if (tipus == 1) sum_kf = sum_kf + (ido*100);           //100 Ft/perc
        else {
            if (tipus == 2) sum_hk = sum_hk + (ido*60);       //60 Ft/perc
            else {
                if (tipus == 3) sum_hb = sum_hb + (ido*40);    //40 Ft/perc
                else printf("Nem definiált típus!");
            }
        }
        i++;
    }
    printf("\nA számla végösszege: %.2f (1) + %.2f (2) + %.2f (3) = %.2f Ft\n", sum_kf,
    sum_hk, sum_hb, sum_kf+sum_hk+sum_hb);
    return 0;
}

/*Többirányú elágazás: if...else if szerkezet */
if (tipus == 1) sum_kf = sum_kf + (ido*100);
else if (tipus == 2) sum_hk = sum_hk + (ido*60);
else if (tipus == 3) sum_hb = sum_hb + (ido*40);
else printf("Nem definiált típus!");

/*Többirányú elágazás: switch utasítás */
switch (tipus) {
    case 1: sum_kf = sum_kf + (ido*100); break;
    case 2: sum_hk = sum_hk + (ido*60); break;
    case 3: sum_hb = sum_hb + (ido*40); break;
    default: printf("Nem definiált típus!");
}

/*Enum (felsorolt típusú) konstansok használatával*/
enum tip {KULFOLDI=1, HALOZATONK, HALOZATONB};
```

```

switch (tipus) {
    case KULFOLDI: sum_kf = sum_kf + (ido*100); break;
    case HALOZATONK: sum_hk = sum_hk + (ido*60); break;
    case HALOZATONB: sum_hb = sum_hb + (ido*40); break;
    default: printf("Nem definiált típus!");
}

```

**5. feladat:** Olvasson be ellenőrzött módon egy intervallumot. Ebben az intervallumban állítson elő automatikusan számpárokat ismétlődés nélkül (azaz (2,3) és (3,2) ugyanaz a pár; és (2,2) nem számpár). Számolja meg és írja ki a párokat.

```

int main() {
    int lower, upper;
    char ch;

    // intervallum ellenőrzött beolvasása, lásd 1. feladat
    // az alsó határ >= 1 legyen, és felsőhatár > alsóhatár

    // számpárok automatikus előállítása ismétlődés nélkül
    int i, j;
    int db = 0;
    for(i=lower; i<=upper; i++) {
        for(j=i+1; j<=upper; j++) {
            printf("(%d, %d) ", i, j);
            db++;
        }
    }
    printf("\nÖsszesen: %d\n", db);
    return 0;
}

```

**6. feladat:** 1 és N között határozza meg a számok szorzatát és mértani átlagát. N-et ellenőrzött módon olvassa be (N>1).

Mértani átlag számítása: a számokat össze kell szorozni és az eredményből annyiadik gyököt kell vonni, ahány átlagolandó szám van.

A szorzatképzéshez használja az összegzés alapalgoritmust.

Hatványozáshoz használja a *math.h* **double pow(double alap, double kitevo)** függvényét.

**7. feladat:** N oldalú négyzet egyik átlója és az alatta/felette lévő terület rajzolása.

```

int main( ) {
    int oldal, sor, oszlop;
    printf("\nAdja meg a negyzet oldalát: ");
    scanf("%d", &oldal);

    /* oszlop==sor átló alatti terület */
    for (sor=1; sor<=oldal; sor++) {
        for (oszlop=1; oszlop<=sor; oszlop++) printf(" * ");
        printf("\n");
    }
    printf("\n");
}

```

```

/* oszlop==sor átló feletti terület */
for (sor=1; sor<=oldal; sor++) {
    for (oszlop=1; oszlop<=sor; oszlop++) printf(" ");
    for (oszlop=sor; oszlop<=oldal; oszlop++) printf(" * ");
    printf("\n");
}
printf("\n");

/* oszlop==sor átló feletti terület (2. verzió) */
for(sor=1; sor<=n; sor++) {
    for(oszlop=1; oszlop<=n; oszlop++) {
        if (oszlop<=sor) printf(" ");
        else printf(" * ");
    }
    printf("\n");
}
printf("\n");

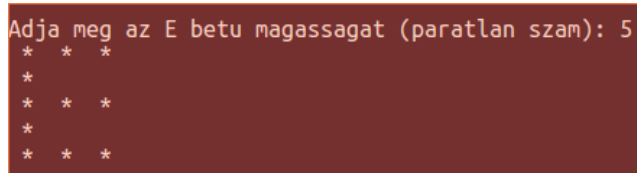
/* oszlop==oldal-sor+1 átló feletti terület */
for (sor=1; sor<=oldal; sor++) {
    for (oszlop=1; oszlop<=oldal-sor+1; oszlop++) printf(" * ");
    printf("\n");
}
printf("\n");

/* oszlop==oldal-sor+1 átló alatti terület */
for (sor=1; sor<=oldal; sor++) {
    for (oszlop=1; oszlop<=oldal-sor+1; oszlop++) printf(" ");
    for (oszlop=oldal-sor+1; oszlop<=oldal; oszlop++) printf(" * ");
    printf("\n");
}
printf("\n");
return 0;
}

```

### Házi feladatok:

1. Rajzoljon ki a képernyőre csillagokból egy E betűt! A betű magasságát kérje be és ellenőrizze, hogy a megadott szám pozitív páratlan szám legyen.



```

Adja meg az E betu magassagat (paratlan szam): 5
* * *
*
* * *
*
* * *

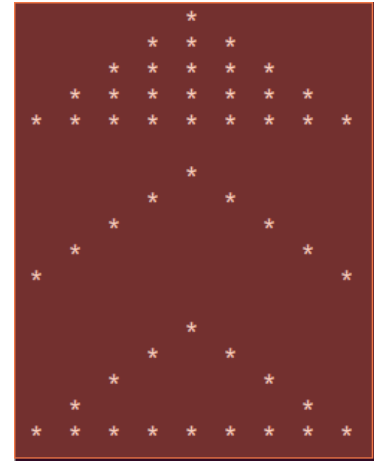
```

2. Az E betű tükrözésével rajzoljon ki 3-ast a képernyőre.

3. Rajzoljon ki # karakterekből egy paralelogrammát. A magasságát és a szélességét olvassa be.

4. Egyenlőszárú, N magasságú háromszög kirajzolása.

```
BE: magasság
sor := 1
AMÍG sor <= magasság ADDIG
    i := 1
    AMÍG i <= magasság-sor ADDIG
        KI: üres karakter
        i := i + 1
    CIKLUS vége
    i := 1
    AMÍG i <= ((sor*2)-1) ADDIG
        KI: *
        i := i+1
    CIKLUS vége
    KI: sortörés
    sor := sor+1
CIKLUS vége
```



5. Módosítsa az előző algoritmust: csak háztetőt, illetve csak a háromszög keretét rajzolja ki.

6. Csúcsára állított egyenlőszárú, N magasságú háromszög kirajzolása.

```
BE: magasság
sor := 1
AMÍG sor <= magasság ADDIG
    i := 1
    AMÍG i <= sor-1 ADDIG
        KI: üres karakter
        i := i + 1
    CIKLUS vége
    i := 1
    AMÍG i <= (magasság*2)-(sor*2)+1 ADDIG
        KI: *
        i := i+1
    CIKLUS vége
    KI: sortörés
    sor := sor+1
CIKLUS vége
```

7. Keresse meg és irassa ki az első N db prímszámot. N értékének beolvasása ellenőrzéssel történjen.

8. Keresse meg és irassa ki 1 és 100 között az összes négyzetszámot.

9. Ellenőrzött adatbeolvasással adjon meg egy intervallumot (alsóhatár < felsőhatár). Majd olvasson be N darab számot és számolja meg, hogy a megadott számok közül hány darab esik az intervallumba.

10. Három megadott számról döntse el, hogy lehetnek-e egy háromszög oldalai.

11. Rajzoljon ki a képernyőre \*-okból a felhasználó által megadott magasságú N és Z betűket. A betűk szélessége egyenlő a megadott magassággal. Végezzen input ellenőrzést: a magasság min. 3.

12. Rajzoljon ki a képernyőre \*-okból a felhasználó által megadott magasságú L, T és H betűket. Végezzen input ellenőrzést: a magasság páratlan szám legyen, legalább 5. A szélességet a program számítja: magasság-2.