Tartalomjegyzék

- 1 10. gyakorlat - C tömbök és karakterláncok
  - 1.1 Feladatok
  - 1.2 C program és C függvény
  - 1.3 Feladat megoldások
    - 1.3.1 percdij
    - 1.3.2 turista
    - 1.3.3 szamok
    - 1.3.4 eratoszthenesz
    - 1.3.5 romai
  - 1.4 Házi feladatok megoldása
    - 1.4.1 ismetles
    - 1.4.2 kaloria

10. gyakorlat - C tömbök és karakterláncok

Ezen a gyakorlaton az el?z? heti el?adás anyagát dolgozzuk fel.

Feladatok

Feladatok a CloudCoder-en megtalálhatóak. Ajánlott sorrend:

- percdij
- turista
- szamok
- eratosztenesz
- romai

C program és C függvény

Ezen a gyakorlaton, és mostantól, vannak olyan feladatok, ahol nem függvényt, hanem teljes programot kell írni. Ez azt jelenti hogy:

- a bemenetet nem függvényparaméterként kapod, hanem *scanf()*-el kell beolvasni
- a kimenetet nem *return*-ölni kell, hanem *printf()*-elni
- kell lennie *main()* függvénynek

A *main()* függvénynek az el?adáson említettel ellentétben *int* kell hogy legyen a visszatérési típusa. (A *void* is m?ködik, de arra panaszkodik a fordító.) És ezért a végén vissza kell térni 0-val (ami egy egész szám ugye). A 0 azt jelenti hogy hibátlanul lefutott a program, tehát az ilyen feladatoknál a *return* mindig 0 legyen, ne az amit kiszámoltatok.

Feladat megoldások

percdij

```
int percdij(int telefon, int hossz) {
    int dijak[] = {20, 35, 12, 7, 78, 44, 31, 19};
    --telefon;
    return dijak[telefon] * hossz;
```

```
}
```

turista

Két lehetséges megoldás is van. Az egyik az, hogy végigmenve a felh?karcolókon, mindegyikre külön ellen?rizzük, hogy lesz-e még utána magasabb, még egy ciklussal. Ennek a kódja:

```
#include <stdio.h>

int main()
{
    int magassag[10];
    int i, j;
    for(i = 0; i < 10; ++i) {
        scanf("%d", &magassag[i]);
    }

    int osszeg = 0;
    for(i = 0; i < 10; ++i) {
        int mindkisebb = 1;
        for(j = i+1; j < 10; ++j) {
            if(magassag[j] >= magassag[i]) {
                mindkisebb = 0;
            }
        }

        if(mindkisebb == 1) {
            osszeg += magassag[i];
        }
    }
    printf("%d", osszeg);

    return 0;
}
```

Van egy másik lehetséges megoldás, amire nehezebb rájönni, de könnyebb leprogramozni. Itt visszafelé megyünk végig a felh?karcolókon, és számontarjuk hogy mekkora volt eddig a legmagasabb. Ha a most következ? alacsonyabb annál, akkor nem látszik, és nem kell beszámítani. Ha magasabb annál, akkor be kell számítani, és mostantól az számít a legmagasabbnak. Kódban ez a megoldás így néz ki:

```
#include <stdio.h>

int main()
{
    int magassag[10];
    int i;
    for(i = 0; i < 10; ++i) {
        scanf("%d", &magassag[i]);
    }

    int max = 0;
    int osszeg = 0;
    for(i = 9; i >= 0; --i) {
        if(magassag[i] > max) {
            osszeg += magassag[i];
            max = magassag[i];
        }
    }
    printf("%d\n", osszeg);

    return 0;
```

```
}
```

szamok

Itt is több lehetséges megoldás van. Szerintem a legegyszer?bb, ha a számok végét keressük csak meg, tehát azokat a helyeket, ahol az aktuális bet? egy számjegy, de a következ? már nem. Erre kód:

```
#include <ctype.h>

int szamok(const char mondat[]) {
  int i;
  int db = 0;
  for(i = 0; mondat[i]; ++i) {
    if(isdigit(mondat[i]) && !isdigit(mondat[i+1])) ++db;
  }

  return db;
}
```

eratoszthenesz

```
#include <stdio.h>

int main()
{
    int N;
    scanf("%d", &N);

    int prime[200];
    int i, j;
    for(i = 0; i < N; ++i) {
      prime[i] = 1;
    }

    for(i = 2; i < N; ++i) {
      if(prime[i]) {
        printf("%d ", i);
        for(j = 2*i; j < N; j+=i) {
          prime[j] = 0;
        }
      }
    }

    return 0;
}
```

romai

Ez nem egy szép megoldás, de megírni nekem ezt volt a legegyszer?bb:

```
#include <stdio.h>

int main()
{
    int N;
    scanf("%d", &N);

    char ki[20];
    int i = 0;
    while(N >= 1000) {
```

```
        ki[i] = 'M';
        ++i;
        N -= 1000;
    }
    if(N >= 900) {
        ki[i] = 'C';
        ki[i+1] = 'M';
        i += 2;
        N -= 900;
    }
    if(N >= 500) {
        ki[i] = 'D';
        i += 1;
        N -= 500;
    }
    if(N >= 400) {
        ki[i] = 'C';
        ki[i+1] = 'D';
        i += 2;
        N -= 400;
    }
    while(N >= 100) {
        ki[i] = 'C';
        i += 1;
        N -= 100;
    }
    if(N >= 90) {
        ki[i] = 'X';
        ki[i+1] = 'C';
        i += 2;
        N -= 90;
    }
    if(N >= 50) {
        ki[i] = 'L';
        i += 1;
        N -= 50;
    }
    if(N >= 40) {
        ki[i] = 'X';
        ki[i+1] = 'L';
        i += 2;
        N -= 40;
    }
    while(N >= 10) {
        ki[i] = 'X';
        i += 1;
        N -= 10;
    }
    if(N >= 9) {
        ki[i] = 'I';
        ki[i+1] = 'X';
        i += 2;
        N -= 9;
    }
    if(N >= 5) {
        ki[i] = 'V';
        i += 1;
        N -= 5;
    }
    if(N >= 4) {
        ki[i] = 'I';
        ki[i+1] = 'V';
        i += 2;
        N -= 4;
```

```
    }
    while(N >= 1) {
        ki[i] = 'I';
        i += 1;
        N -= 1;
    }
    ki[i] = 0;
    printf(ki);

    return 0;
}
```

Egy rövidebb megoldás, ami viszont használ mutatókat tartalmazó tömböt, és az strcat() függvényt a *string.h* beépített könyvtárból.:

```
#include <stdio.h>
#include <string.h>

#define ERTEKEK_DB 13

int main()
{
    int N;
    scanf("%d", &N);

    const char *nevek[ERTEKEK_DB] = {"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "
    int ertekek[ERTEKEK_DB] =       {1000, 900, 500,  400, 100,   90,  50,   40,  10,    9,   5,

    char ki[20];
    ki[0] = 0;
    int i;

    for(i = 0; i < ERTEKEK_DB; ++i) {
        while(N >= ertekek[i]) {
            N -= ertekek[i];
            strcat(ki, nevek[i]);
        }
    }
    printf(ki);

    return 0;
}
```

Házi feladatok megoldása

ismetles

```
#include <stdio.h>

int main() {
    int a, b, i;
    scanf("%d%d", &a, &b);
    char ki[20];
    char betu = 'a' - 1 + a;
    for(i = 0; i < b; ++i){
        ki[i] = betu;
    }
    ki[b] = 0;
    printf(ki);

    return 0;
```

```
}
```

kaloria

```
#include <stdio.h>

int main() {
  int n, k;
  int sutik[20];
    int i;
    scanf("%d", &n);
    for(i = 0; i < n; ++i) {
      scanf("%d", &sutik[i]);
    }
    scanf("%d", &k);

    int ossz = 0;
    for(i = 0; i < n; i += k) {
      ossz += sutik[i];
    }

    printf("%d\n", ossz);
}
```