Dolgozatok:

Írjon programot, amely bekér három valós (lebegőpontos) számot, majd kiírja a megadott számok közül, hogy melyik a legnagyobb és a legkisebb.

Megoldás:

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

int main(){
	setlocale (LC_ALL, "");
	float a, b, c, nagy, kicsi;

	printf("Kérem az elso számot:"); scanf("%f", &a);
	printf("Kérem a második számot:"); scanf("%f", &b);
	printf("Kérem a harmadik számot:"); scanf("%f", &c);

	nagy = kicsi = a;
	if(b>nagy) nagy=b;
	else if(b<kicsi) kicsi=b;
	if(c>nagy) nagy=c;
	else if(c<kicsi) kicsi=c;

	printf("A legnagyobb szám: %10.2f\n", nagy);
	printf("A legkisebb szám:  %10.2f\n", kicsi);
}
```

Készítsen programot, amely meghatározza a víz halmazállapotát és kiírja azt („jég”, „víz”, „gőz”). A hőmérséklet értékeket (°C-ban) valós (lebegőpontos) számként kérje be. Ügyeljen arra, hogy az abszolút 0 foknál (-273,16°C) hidegebb semmi sem lehet. (Hibás adatmegadás esetén adjon hibajelzést!)

Megoldás:

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

int main(){
	setlocale(LC_ALL, "hun");
	float hom;

	printf("Kérem a hőmérsékletet (C fok): "); scanf("%f",&hom);

	if (hom < -273.16) printf("Hibás adat.\n");
	else if ( hom < 0) printf("Jég\n");
		else if (hom < 100) printf("Víz\n");
			else printf("Gőz\n");
}
```

Írjon programot, amely bekér három érdemjegyet, majd kiszámítja az osztályátlagot egészre kerekítve (1-től 5-ig egész szám!), ezután az eredményt szövegesen kiírja (5: „jeles”; 4: „jó”; 3: „közepes”; 2: „elégséges”, 1: „elégtelen”). (Hibás adatmegadás esetén adjon hibajelzést!)

Megoldás:

```
#include <stdio.h>
#include <iostream>
int main(){
	setlocale(LC_ALL, "hun");
	int a, b, c, atlag;
	printf("Kérek egy érdemjegyet: "); scanf("%d",&a);
	printf("Kérek egy érdemjegyet: "); scanf("%d",&b);
	printf("Kérek egy érdemjegyet: "); scanf("%d",&c);
	if(a<1 || a>5 || b<1 || b>5 || c<1 || c>5) printf("Hibás adatot adott meg!");
	else {
		atlag=(a+b+c)/3;
		switch(atlag){
			case 1: printf("elégtelen\n"); break;
			case 2: printf("elégséges\n"); break;
			case 3: printf("közepes\n"); break;
			case 4: printf("jó\n"); break;
			case 5: printf("jeles\n"); break;
		}
	}

}
```

Vezérlési szerkezetek (ciklusok):

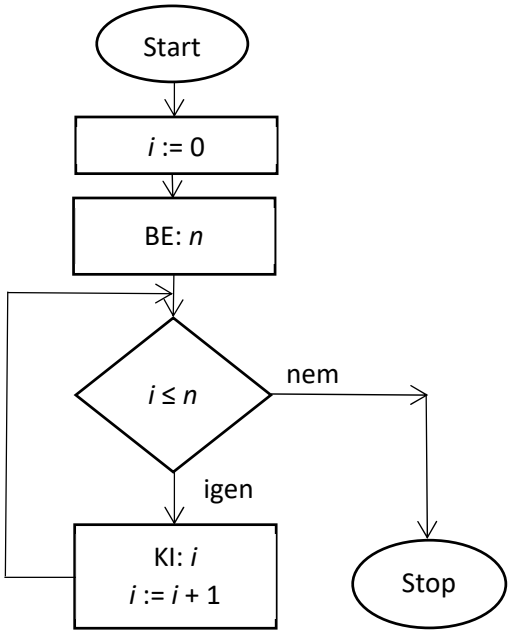
1. Az elöltesztelő ciklus (while).

Példa:

Olvassunk be egy tetszőleges egész számot, majd írjuk ki 0-tól a megadott értékig az egész számokat.

Megbeszélés:

1. A változók kezdőértékének megadása, annak fontossága.
2. Az elöltesztelő ciklus tulajdonságai, részei (ciklusfej, ciklusmag, ciklusváltozó).
3. Az inkrementáló és dekrementáló operátor (++, --), (pl: i = i + 1; → i++;).



Megoldás:

```
#include <stdio.h>
int main(){
        int n, i = 0;
        printf("Kérek egy számot: "); scanf("%d", &n);
        while (i <= n){
                printf("%d\n", i);
                i++;
        }
}
```

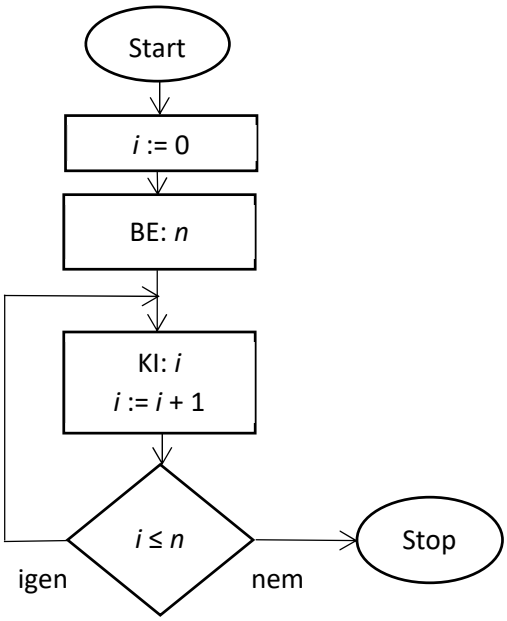
2. A hátultesztelő ciklus (do while).

Példa:

Az előző feladatot készítsük el hátultesztelő ciklussal.

Megbeszélés:

1. A hátultesztelő ciklus tulajdonságai, részei (ciklusfej, ciklusmag, ciklusváltozó).
2. Elöl- és hátultesztelő ciklus összehasonlítása. Mi történik, ha adatmegadáskor negatív számot adunk meg?



Megoldás:

```
#include <stdio.h>
int main(){
        int n, i = 0;
        printf("Kérek egy számot: "); scanf("%d", &n);
        do{
                printf("%d\n", i);
                i++;
        } while (i <= n);
}
```

3. A számláló ciklus (for).

Példa:
Az előző feladatot készítsük el számláló ciklussal.

Megbeszélés:
1. A számláló ciklus tulajdonsága, részei.
2. Kezdő értékadás(ok); feltétel; vég értékadás(ok).
3. Blokk, blokkon belüli változó deklarálás. Mi történik, ha blokkon kívül is van egy azonos nevű változó?
4. A , (vessző) operátor.
5. Az elkészült program alapján a hallgatók döntsék el, hogy elöl- vagy hátultesztelő-e a ciklus.

Megoldás:

```
#include <stdio.h>
int main(){
        int n, i;
        printf("Kérek egy számot: "); scanf("%d",&n);
        for( i=0; i <= n; i++)
                printf("%d\n", i);
}
```

Példa:
Kérjünk be egész számokat „0” végjelig, számítsuk ki és írjuk ki a számok összegét, átlagát és az átlag számításnál keletkező maradékot.

Megbeszélés:
1. Összeg kiszámítása.
2. Értékadó operátorok: =, +=, -=, *=, /=, %=, (pl: a = a + b; → a += b;)

Megoldás:

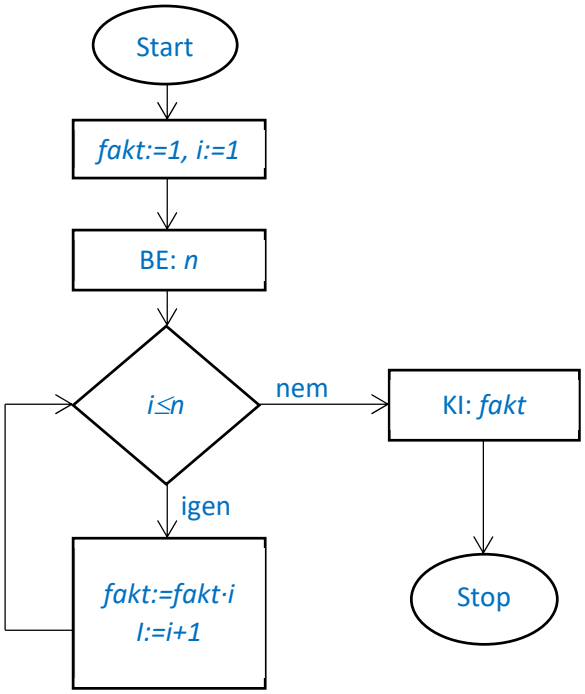
```
#include <stdio.h>
#include <iostream>
int main(){
    setlocale (LC_ALL, "");
    int szam, n=0, osszeg=0;
    printf("Kérek egy számot: "); scanf("%d", &szam);
    while(szam){
        osszeg += szam;
        n++;
        printf("Kérek egy számot: "); scanf("%d", &szam);
    }
    printf("A számok összege = %d, az átlag = %d, a maradék = %d\n", osszeg, osszeg/n, osszeg%n);
}
```

Példa:

A korábban megismert faktoriális számító algoritmus segítségével készítsük el a programot.

Megbeszélés:

1. $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$, ha $n \in \mathbb{N}, n \neq 0$, és $0! = 1$.
2. n!=(n-1)!·n,



Megoldás:

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

int main(){
	setlocale (LC_ALL, "");
	int n, fakt=1;
	printf("Kérek egy számot: "); scanf("%d",&n);
	for(int i = 1; i <= n; i++)
		fakt=fakt * i;
	printf("%d! = %d\n", n, fakt);

}
```

Karakter típus:

A betűket, a számjegyeket, az írásjeleket és az egyéb jeleket, összefoglaló néven karaktereknek hívjuk. Az egy vagy több karakterből álló sorozatot szövegnek (sztringnek) nevezzük.

A számítógépen a karaktereket kódolt formában, egész számként tároljuk, azaz nem a karakter képét, hanem a karakterek numerikus kódjait. Megjelenítéskor a tárolt kódhoz kapcsoljuk hozzá azt az információt, hogy milyen formában (stílusban) jelenjen meg a karakter. A karaktereket és kódjukat különböző szabványosított táblázatok tartalmazzák. A legelterjedtebb az Amerikai Szabványügyi Hivatal (ANSI), 1977-ben megalkotott ASCII kódtáblája, amely 128 karaktert (vezérlőkarakter, betű, számjegy, írásjel) tartalmaz. A 128 karakter 7 biten ábrázolható, a nyolcadik bitet paritásbitként, ellenőrzésre használták. Később a számítástechnika nemzetközivé válásával szükségessé vált – az angol karakterkészletben nem található – különböző nemzeti karakterek bevezetése. Ennek érdekében 8 bitre bővítették a kódtáblát. (Legelterjedtebb 8 bites szabvány az ISO 8859 (Latin) karakterkészlet.) Mivel az összes nemzeti írásjel tárolására ez sem volt elég, több különböző, nemzeti kódtáblát alakítottak ki (például: nyugat-európai, kelet-európai, cirill stb.). Természetesen ezek a kódtáblák tartalmazzák az eredeti 7 bites US-ASCII kódokat.

(Manapság egyre gyakrabban használják a Unicode szabványt, az UTF-8, UTF-16 vagy az UTF-32 karakterkódolást.)

Példa:
Olvassunk be egy tetszőleges karaktert és döntsük el, hogy az szám karakter-e.

Megbeszélés:
4. A karaktertípusú változó deklarálása (*char*).
5. Karakter konstansok: ’A’, ’5’, ’\n’, ’\065’.

Megoldás:

```
#include <stdio.h>
#include <iostream>
int main(){
        setlocale (LC_ALL, "");
        char c;
        printf("Kérek egy karaktert: "); scanf("%c",&c);
        if(c>='0' && c<='9') printf("szám\n");
        else printf("nem szám\n");
}
```

Példa:

Írjunk programot, amely karaktereket olvas be (ENTER-ig) a billentyűzetről, és visszaírja a képernyőre. Használjuk a *getchar()* és a *putchar()* függvényeket.

Megbeszélés:

1. A *putchar()* függvény.
2. Billentyű puffermemória.

Megoldás:

```
#include <stdio.h>
#include <iostream>
int main(){
        setlocale (LC_ALL, "");
        char c;
        printf("Kérek egy karaktersorozatot, ENTER-ig:\n");
        c=getchar();
        while(c!='\n'){
                putchar(c);
                c=getchar();
}       }
```

Példa:

Módosítsuk az előző programot úgy, hogy EOF végjelig olvasson a program.

Megbeszélés:

1. EOF konstans.

Megoldás:

```
#include <stdio.h>
#include <iostream>
int main(){
        setlocale (LC_ALL, "");
        char c;
        printf("Kérek egy karaktersorozatot, EOF-ig:\n");
        while((c=getchar())!=EOF)
                putchar(c);
}
```

Feladatok:

1. A program kérjen be egész számokat tetszőleges végjelig. A beolvasás után írja ki a megadott számok közül a legkisebb értéket. A feladatot elöl tesztelő ciklussal oldja meg!
2. A program kérjen be tetszőleges számú életkort, 0 (nulla) végjelig. Számolja meg, hogy hány 30-59 év közötti életkort adtunk meg. A feladatot hátul tesztelő ciklussal oldja meg!
3. A program kérjen be karaktereket EOF végjelig. Határozza meg, hogy hány darab kisbetű karaktert adtunk meg. A feladatot hátul tesztelő ciklussal oldja meg!
4. A program kérjen be osztályzatokat, számítsa ki az átlagot, majd az eredményt szövegesen írja ki. (4,5 felett - „jeles”; 3,5 – 4,5 - „jó”; 2,5 – 3,5 - „közepes”; 2 – 2.5 - „elégséges”, 2 alatt - „elégtelen”).
5. A program kérjen be karaktereket EOF végjelig. Határozza meg, hogy hány darab nem szám karaktert adtunk meg. A feladatot elöl tesztelő ciklussal oldja meg!
6. A program kérjen be tetszőleges számú hőmérsékletet, 0 (nulla) végjelig. Határozza meg, hogy hányszor adtunk meg 0 fok alatti hőmérsékletet.
7. Készítsen programot, amely tetszőleges számú téglalap területét számítja ki. A számolást a program addig végezze, amíg bármelyik bemenő adat nem nulla. A feladatot elöl tesztelő ciklussal oldja meg!
8. A program kérjen be karaktereket EOF végjelig. Határozza meg, hogy hány darab nem betű karaktert adtunk meg.
9. A program kérjen be egész számokat tetszőleges végjelig. A beolvasás után írja ki a megadott számok közül a legnagyobb értéket. A feladatot hátul tesztelő ciklussal oldja meg!
10. Készítsen programot, amely tetszőleges számú hasáb térfogatát számítja ki. A számolást a program addig végezze, amíg bármelyik bemenő adat nem nulla. A feladatot hátul tesztelő ciklussal oldja meg!
11. A program kérjen be karaktereket EOF végjelig, majd írja ki azokat. Kiíráskor hagyja ki a soremelés karaktereket.
12. A program kérjen be egy sor szöveget (karakterenkén), majd betűnként külön sorba írja ki azt.
13. A program kérjen be karaktereket EOF végjelig, majd írja ki azokat szavanként külön sorba.
14. Olvasson be végjelig hónap sorszámokat. A program írja ki, a hónapok nevét, valamint számolja meg és írja ki, hogy hány nyári hónapot adtunk meg. Hibás adatmegadás esetén adjon hibajelzést!