Dolgozatok:

Olvasson be egy sor szöveget egy sztringbe, majd egy karaktert. Írja ki, hogy a megadott karakter hol fordul elő a sztrinben (hányadik betű), az összes előfordulását írja ki!

Megoldás:

```
#include <stdio.h>
#include <iostream>
int main(){
        setlocale(LC_ALL, "hun");
        char szoveg[80], c;
        printf("Kérek egy sor szöveget: "); scanf("%[^\n]", szoveg);
        fflush(stdin);
        printf("Kérek egy karaktert: "); scanf("%c", &c);
        printf("A karakter helye: ");
        for (int i=0; szoveg[i]; i++)
                if (szoveg[i] == c) printf("%d. ", i+1);
}
```

Olvasson be egy sor szöveget egy sztringbe, majd számolja meg, hogy a sztringben hány kisbetű, nagybetű, szám és egyéb karaktert található.

Megoldás:

```
#include <stdio.h>
#include <iostream>
int main(){
    setlocale(LC_ALL, "hun");
    char  szoveg[80], *s=szoveg, i=0,szam=0,kisb=0, nagyb=0, egyeb=0;
    printf("Kérek egy sor szöveget: "); scanf("%[^\n]", szoveg);
    while(*s){
            if (*s>='0' && *s<='9') szam++;
            else if (*s>='a' && *s <='z') kisb++;
                else if (*s>='A' && *s <='Z') nagyb++;
                    else egyeb++;
    }
    printf("Szám:\t\t%d\nKisbetű:\t%d\nNagybetű:\t%d\nEgyéb:\t\t%d\n", szam, kisb, nagyb, egyeb);
}
```

Egy tömbnek kezdőértéként adja meg az évszakok neveit. Majd kérje be egy hónap sorszámát. Írja ki a hónap számához tartozó évszakot (az évszakokat tartalmazó tömb segítségével). Hibás adatmegadás esetén írjon ki hibaüzenetet!

Megoldás:

```
#include <stdio.h>
#include <iostream>
int main (){
        char *evszak[]={
                        "Tavasz",
                        "Nyár",
                        "Ősz",
                        "Tél",
                        "Hibás hónapszám"
                      };
        int ho;
        setlocale(LC_ALL, "hun");
        printf("kerek egy hónapszámot:");scanf("%d",&ho);
        switch (ho){
                case 3: case 4: case 5: printf("%s\n",evszak[0]); break;
                case 6: case 7: case 8: printf("%s\n",evszak[1]); break;
                case 9: case 10: case 11: printf("%s\n",evszak[2]); break;
                case 1: case 2: case 12: printf("%s\n",evszak[3]); break;
                default: printf("%s\n",evszak[4]); break;
        }
}
```

Megoldás:

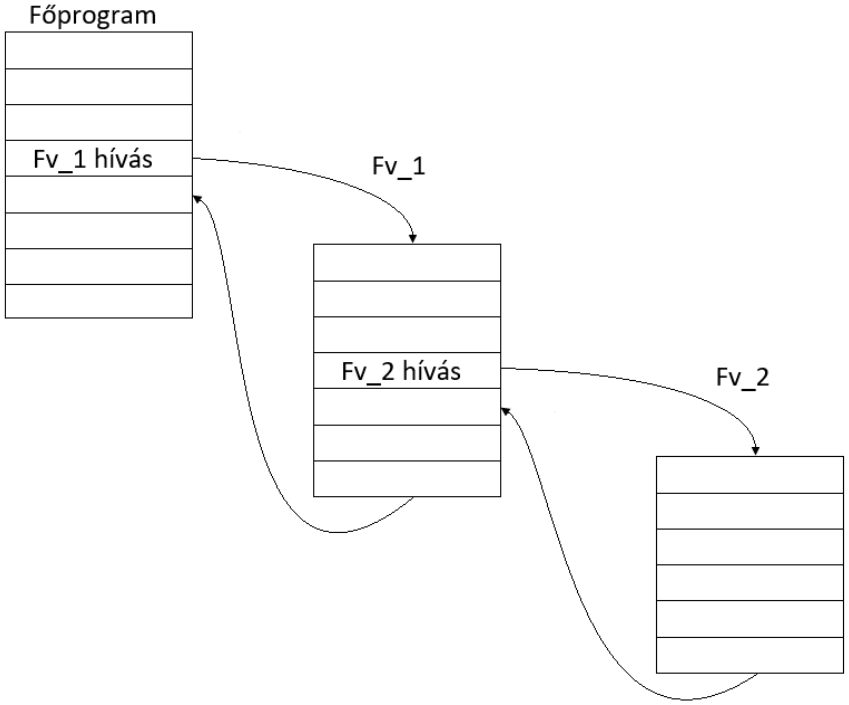
```
#include <stdio.h>
#include <iostream>
int main (){
        char *evszak[]={
                        "Tavasz",
                        "Nyár",
                        "Ősz",
                        "Tél",
                      };
        int ho;
        setlocale(LC_ALL, "hun");
        printf("kerek egy hónapszámot:"); scanf("%d",&ho);
        if(ho < 1 || ho > 12) printf("Hibás hónap\n");
        else if(ho == 1 || ho == 2 || ho == 12) printf("%s\n", evszak[3]);
                else if(ho < 6) printf("%s\n", evszak[0]);
                        else if(ho < 9) printf("%s\n", evszak[1]);
                                else printf("%s\n", evszak[2]);
}
```

Függvények

A függvény a programunk egy viszonylag zárt, független része, amely a bemeneti adatok felhasználásával előállítja a kimenetén megjelenő értéket. Függvényeket általában akkor használunk: ha egy feladatrészt többször kell végrehajtanunk; ha olyan részfeladatot készítünk, amelyet más programban is fel akarunk használni; vagy ha ezzel rendezettebbé, átláthatóbbá tehetjük a programunkat. A függvényeknek van nevük (azonosítójuk), lehetnek paraméterei és visszaadott értéke.

A működésük hasonló a matematikában megismert függvényekhez: *y=f(x)*, ahol *x* a független változó (argument) és *y* a függő változó (függvény érték). Programozásban a független változót paraméternek nevezzük. Egy függvény lehet paraméter nélküli és több paraméteres is. A C nyelvben csak értékparaméter van! Függvényérték csak egy lehet!

A függvények hívása és az azokból való visszatérés, feltételnélküli vezérlésátadások.



A függvény működése:

- A híváskor megadott aktuális paraméterek értékét elmenti a verembe.
- A következő utasítás címét elmenti a verembe, erre a címre (a hívást követő utasításra) tér vissza a függvény befejezése után.
- Átadódik a vezérlés.
- A veremben tárolt értékekhez formális paramétert rendel (ezek nevével hivatkozhatunk a tárolt adatokra).
- A függvényben megadott utasítások végrehajtása után, vagy a *return* utasítás hatására visszatér a híváskor elmentett címre. A visszatéréssel egyidőben a verem állapotát visszaállítja a hívást megelőzőre. Vagyis a veremben tárolt – a függvényben használt adatok – „elvesznek”.

1. Érték átadás:

A hívó programból az adatokat paramétereken keresztül (a veremben) adjuk át a függvénynek. A függvényben deklarált (helyi vagy lokális) változók a formális paraméterekkel együtt, szintén a veremben helyezkednek el. A műveletek eredményei is a verembe kerülnek. Mivel visszatéréskor a verem a hívást megelőző állapotra áll vissza, ezért a függvényben használt össze lokális adat elveszik.

Abban az esetben, ha a függvényben kiszámolt eredményt vissza szeretnénk adni a hívó programnak, akkor azt függvény értékként tehetjük meg a return eredmény utasítással.

Példa:

Számítsuk ki egy téglalap területét és kerületét függvények segítségével. Az oldalak hosszát a *main()* kérje be, a területet és kerületet függvénnyel számítsuk ki. A függvények paraméterei a téglalap oldalhosszai, a visszaadott értékek a kerület illetve terület értéke, amit a *main()* ír ki.

Megbeszélés:

1. Függvény deklarálása, létrehozása. Függvény neve, típusa, értéke, paraméterek.
2. Aktuális és formális paraméterek. Verem szerepe a függvényhívásban.
3. Érték szerinti adatátadás.
4. Lokális változó.

Megoldás:

```
#include <stdio.h>
int kerulet(int, int);
int terulet(int, int);
int main(){
        int a, b, ter, ker;
        printf("Kérem a téglalap egyik oldalának hosszát: "); scanf("%d", &a);
        printf("Kérem a téglalap egyik oldalának hosszát: "); scanf("%d", &b);
        ker = kerulet(a, b);
        ter = terulet(a, b);
        printf("A téglalap kerülete=%d, a területe= %d\n", ker, ter);
}
int kerulet(int x, int y){
        int z;
        z = 2*(x+y);
        return z;
}
int terulet(int x, int y){
        return x*y;
}
```

2. Cím átadás:

Abban az esetben, ha több értéket is kiszámoltatunk a függvénnyel, és azt vissza szeretnénk adni a hívó programnak, akkor ezt nem tehetjük meg függvényértékként (a függvénynek csak egy adat lehet). Ilyenkor paraméter segítségével adjuk vissza az értéket. Ez csak úgy lehetséges, ha paraméterként nem a változó értékét adjuk át, hanem annak a címét, és a függvényben az adott címre pakoljuk a kiszámított értéket.

Példa:

Alakítsuk át az előző programot úgy, hogy egy függvényt használjunk. A függvény paraméterei a téglalap oldalhosszai, valamint a kerület és terület, visszaadott érték nincs. A kerület illetve terület értékét a *main()* írja ki.

Megbeszélés:

1. Cím szerinti adatátadás.

Megoldás:

```
#include <stdio.h>
void tegla(int, int, int*, int*);
int main(){
        int a, b, ter, ker;
        printf("Kérem a téglalap egyik oldalának hosszát: "); scanf("%d", &a);
        printf("Kérem a téglalap egyik oldalának hosszát: "); scanf("%d", &b);
        tegla(a, b, &ker, &ter);
        printf("A téglalap kerülete=%d, a területe= %d\n", ker, ter);
}
void tegla(int x, int y, int *k, int *t){
        *k = 2*(x+y);
        *t = x*y;
}
```

3. Globális változók használata:

A változókat csoportosíthatjuk hatáskör és élettartam szerint is:

1. Abban az esetben, ha egy változót blokkon vagy függvényen belül hozunk létre, akkor helyi (lokális) változóról beszélünk. Ekkor a változó hatásköre csak arra a blokkra (függvényre) érvényes, ahol létrehoztuk. Élettartalma addig szól, amig az utasításvégrehajtás a blokkon (függvényen) belül van, tartalma utána „elveszik”.
2. A változó globális, ha az a program egész területén látható és módosítható. Tehát hatóköre az egész program (függvények is), élettartama a program végéig tart.

Példa:

Alakítsuk át az előző programot úgy, hogy globális változókat használunk.

Megbeszélés:

1. Globális változó létrehozása.
2. Globális változók használatának következménye, miért ne használjuk.

Megoldás:

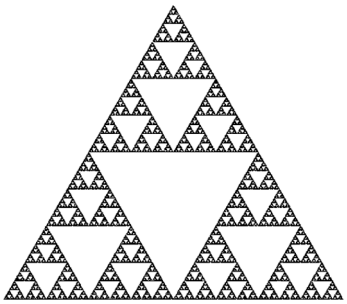
```
#include <stdio.h>
void tegla();
int a, b, ter, ker;
int main(){
        printf("Kérem a téglalap egyik oldalának hosszát: "); scanf("%d", &a);
        printf("Kérem a téglalap egyik oldalának hosszát: "); scanf("%d", &b);
        tegla();
        printf("A téglalap kerülete=%d, a területe= %d\n", ker, ter);
}
void tegla(){
        ker = 2*(a + b);
        ter = a * b;
}
```

Rekurzió:

Matematikában olyan eljárás (műveletsor) amely önmagát hívja meg. Tipikus példa a rekurzív sorozatok, amikor a sorozat tetszőleges tagját az előtte állók függvényében kapjuk meg. Ilyen például a Fibonacci-számok (sorozat):

$$f_n=f_{n-1}+f_{n-2},\ f_1=1,\ f_2=1$$

Önmagát ismétlő egyenlettel leírt geometriai forma, például a fraktálok.



A nyelvtudományban is használat fogalom, itt a tagmondatok egymásba ágyazását vagy összetett szavak képzését értjük.

Informatikában a rekurzión egy programozási módszert értünk, amikor egy függvény önmagát hívja meg. Természetesen gondoskodnunk kell arról, hogy véges számú lépés után leállítsuk a folyamatot. Ahogy korábban láttuk, a függvény használatakor a visszatérési cím, a paraméterek és a lokális változók a veremben helyezkednek el. A függvény ismételt meghívásakor ezek újra és újra a verembe kerülnek. Így az egymásba ágyazások számát (rekurzió mélységét) a verem mérete korlátozza.

Példa:

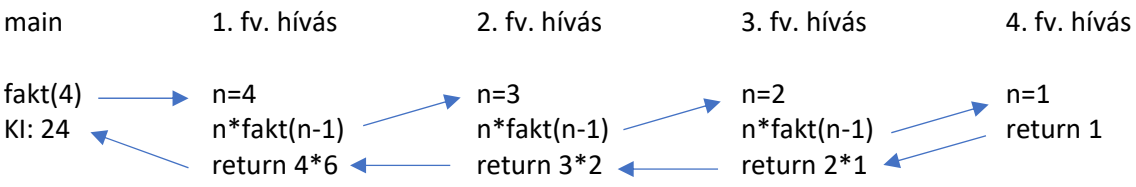
Készítsük el a már korábban megbeszélt faktoriális számító programunkat rekurzióval.

Emlékeztető:

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>
int fakt(int);
int main(){
        int n;
        printf("Kerek egy szamot: "); scanf("%d",&n);
        printf("%d! = %d\n", n, fakt(n));
}
int fakt(int n){
        if(n<=1) return 1;
        return n*fakt(n-1);
}
```



Példa:

Számítsuk ki rekurzióval a fibonacci sorozat tetszőleges tagját.

Emlékeztető:

1. $fn = fn - 1 + fn - 2, f1 = 1, f2 = 1$

Megoldás:

```
#include <stdio.h>
int fibonacci(int);
int main(){
        int n;
        printf("Kerek egy szamot: "); scanf("%d",&n);
        printf("A %d. fibonacci szam = %d\n", n, fibonacci(n));
}
int fibonacci(int n) {
        if ( (0 == n) || (1 == n) )
                return n;
        else
                return fibonacci(n-1) + fibonacci(n-2);
}
```

Példa:

Függvény segítségével határozzuk meg egy sornyi szöveg hosszát.

Megoldás:

```
#include <stdio.h>
int strlen(char *s){
        char *p=s;
        while(*s)s++;
        return s-p;
}
/*
int strlen(char s[]){
        int i=0;
        while(s[i])i++;
        return i;
}
*/
int main(){
        char in[80], i=0;
        printf("Kerem a szoveget: "); scanf("%[^\n]", in);
        printf("A szoveg hossza = %d\n",strlen(in));
}
```

Példa:

Függvény segítségével másoljunk át egy szöveget.

Megoldás:

```
#include <stdio.h>
void strcpy(char *s, char *t){
        while(*s++=*t++);
}
int main(){
        char in[80], out[80], i=0;
        printf("Kerem a szoveget: "); scanf("%[^\n]", in);
        strcpy(out, in);
        printf("\n%s\n",out);
}
```

Példa:

Függvény segítségével határozzuk meg egy hónap nevét.

Megbeszélés:

1. Feltételes értékadás:

Határozzuk meg két szám közül a kisebbet:

```
if(a < b)
        z=a;
else
        z=b;
```

helyette:

```
z = (a < b)? a : b;
```

Megoldás:

```
#include <stdio.h>
char *honev(int ho){
        static char *nev[]={
                "hibas honap",
                "januar",
                "februar",
                "marcius",
                "aprilis",
                "majus",
                "junius",
                "julius",
                "augusztus",
                "szeptember",
                "oktober",
                "november",
                "december"
                };
        return (ho<1 || ho>12)? nev[0]:nev[ho];
}
int main(){
        int ho;
        printf("Kerem a honap szamat: "); scanf("%d",&ho);
        printf("\n%s\n",honev(ho));
}
```

Feladatok:

1. Készítsen programot, amely **függvény** segítségével beolvas egy karaktersort (stringet). A beolvasott karakterek közül megszámolja a kisbetű karaktereket. A karakterek számát átadja a **main**-nek, amely kiírja azt.
2. Készítsen programot, amely **függvény** segítségével beolvas egy karaktersort (stringet). A beolvasott karakterek közül megszámolja a szám karaktereket. A karakterek számát átadja a **main**-nek, amely kiírja azt.
3. Készítsen programot, amely **függvény** segítségével beolvas egy szövegsort, majd meghatározza a szavak számát. A szavak számát átadja a **main**-nek, amely kiírja azt.
4. Készítsen programot, amely **függvény** segítségével kiszámítja egy hasáb térfogatát. A hasáb adatait a **main** kéri be, azokat paraméterként átadja a függvénynek. A **függvény** kiszámítja a térfogatot, majd visszaadja azt. Az eredményt a **main** írja ki.
5. Készítsen programot, amely **függvény** segítségével meghatározza, hogy a megadott oldalhosszakból szerkeszthető-e háromszög (bármely két oldal összege nagyobb a harmadiknál). Az oldalak hosszát a **main** kéri be, azokat paraméterként átadja a **függvénynek**. A **függvény** meghatározza a szerkeszthetőséget és ennek megfelelően egy logikai értékkel tér vissza. Az eredményt a **main** írja ki.
6. Készítsen programot, amely **függvény** segítségével meghatározza három szám közül a legkisebbet. Az adatokat a **main** kéri be, azokat paraméterként átadja a **függvénynek**. A **függvény** meghatározza a legkisebb értéket, majd ezzel tér vissza. Az eredményt a **main** írja ki.
7. Készítsen programot, amely **függvény** segítségével meghatározza a víz halmazállapotát és kiírja azt („jég”, „víz”, „gőz”). A hőmérsékletet a **main** kéri be, azt paraméterként átadja a függvénynek. A **függvény** meghatározza, majd kiírja a megfelelő szöveget.
8. Készítsen programot, amely **függvény** segítségével meghatározza három szám átlagát. Az adatokat a **main** kéri be, azokat paraméterként átadja a **függvénynek**. A **függvény** meghatározza az átlagot, majd ezzel tér vissza. Az eredményt a **main** írja ki.
9. Készítsen programot, amely **függvény** segítségével lerendezi egy tömb elemeit. A tömb értékeit a **main** kéri be, majd a rendezett tömböt kiírja. **A függvényben mutatókat használjon!**
10. Készítsen egy kereső **függvényt**, amely egy szövegben megkeresi egy tetszőleges karakter első előfordulását. A szöveget és egy karaktert a **main** kéri be, ezeket átadja a függvénynek, amely megkeresi, hogy a karakter hányadik a szövegben. A karakter sorszámát átadja a **main**-nek, amely kiírja azt.
11. Készítsen programot, amely **függvény** segítségével meghatározza egy szövegsor hosszát. A függvény paramétere a szöveg, visszaadott érték a hossz. A karakterek számát a **main**, írja ki. A feladat megoldásához mutatókat használjon!
12. Készítsen programot, amely **függvény** segítségével kiszámítja egy téglatest felszínét és térfogatát. A téglatest adatait a **main** kéri be, azokat paraméterként átadja a függvénynek. A **függvény** kiszámítja a felszínt és a térfogatot, majd paraméterek segítségével visszaadja azokat. Az eredményt a **main** írja ki.

13. Készítsen programot, amely **függvény** segítségével meghatározza, hogy a megadott oldalhosszakból szerkeszthető-e háromszög (bármely két oldal összege nagyobb a harmadiknál). Az oldalak hosszát a **main** kéri be, azokat paraméterként átadja a **függvénynek**. A **függvény** meghatározza a szerkeszthetőséget és ennek megfelelően egy szöveg-mutatóval (*szerkeszthető*, *nem szerkeszthető*) tér vissza. Az eredményt a **main** írja ki.
14. Készítsen programot, amely **függvény** segítségével meghatározza három szám sorrendjét. Az adatokat a **main** kéri be, azokat paraméterként átadja a **függvénynek**. A **függvény** a paramétereket értékeit úgy módosítja, hogy azok nagyságrendben legyenek. Az eredményt a **main** írja ki.
15. Készítsen programot, amely **függvény** segítségével meghatározza a víz halmazállapotát. A hőmérsékletet a **main** kéri be, azt paraméterként átadja a függvénynek. A **függvény** a megfelelő (*jég, víz, gőz*) szöveg-mutatóval tér vissza, amit a **main** ír ki.
16. Készítsen programot, amely **függvény** segítségével meghatározza az évszakok nevét. A hónap sorszámát a **main** kéri be, azt paraméterként átadja a függvénynek. A **függvény** a megfelelő (*tavasz, nyár, ősz, tél*) szöveg-mutatóval tér vissza, amit a **main** ír ki.