Miskolci Egyetem
Általános Informatikai Intézeti Tanszék

Nappali tagozat
Őszi félév

GEIAL311-B
Programozás alapjai
Mérnök- / Programtervező / Gazdaságinformatikus alapszak (BSc)

A tárgy előadója, leckekönyvi jegyzője: Dr. Baksáné Dr. Varga Erika, egyetemi docens
A tárgy lezárásának módja: aláírás és vizsga
Mintatanterv szerinti félév: 1
Kredit: 5
Kontakt órák száma / hét: 2 előadás, 1 tantermi gyakorlat, 2 labor gyakorlat

ÜTEMTERV

Hét	Előadás	Tantermi gyakorlat	Labor gyakorlat
1.	A számítógép programozása. Programozási nyelvek. A programkészítés menete. Integrált fejlesztő rendszerek.	A C program szerkezete. Egyszerű ki- és bemeneti függvények C-ben.	Labor használati szabályzat ismertetése. Felhasználói account adminisztráció. Ismerkedés a CodeBlocks fejlesztő környezettel. Első C programok.
2.	A C programozási nyelv jellemzői és szintaktikai egységei.	Algoritmizálás gyakorlat	Első számítást végző C programok.
3.	Operátorok és kifejezések, típuskonverzió. A C nyelv szelekciós és ciklus utasításai.	Vezérlési szerkezetek C nyelvi megvalósítása	Programozási feladatok az egyszerű vezérlési szerkezetek gyakorlására.
4.	A C nyelv utasításai II.	Alapalgoritmusok I. Példák összegzésre, számlálásra.	Programozási feladatok az egymásba ágyazott vezérlési szerkezetek gyakorlására.
5.	Egydimenziós tömbök és mutatók. Véletlenszám generálás és változatai.	Alapalgoritmusok II. Példák eldöntésre, kiválasztásra, szélsőérték kiválasztásra.	Tömbkezelés, alapalgoritmusok C nyelvi megvalósítása.
6.	Sztringek és kezelő függvényeik.	Alapalgoritmusok III. Példák keresésekre.	Sztringkezelés, alapalgoritmusok C nyelvi megvalósítása.
7.	Alapalgoritmusok IV. Rendezések.	Alapalgoritmusok V. Tömbi algoritmusok megvalósítása.	**1. programozás számonkérés (tömbkezelés, alapalgoritmusok I-III)**
8.	Függvények, programtervezési alapelvek.	Top-down program-tervezésre példák. Euklidészi algoritmus.	Függvényírás, top-down programtervezés.

Hét	Előadás	Tantermi gyakorlat	Labor gyakorlat
9.	Rekurzív függvények definiálása, használati esetei.	Számelméleti algoritmusok.	Függvényírás, top-down programtervezés.
10.	Tárolási osztályok. Moduláris programozás.	Több modulos program, saját header állomány készítése.	Saját függvénykönyvtár létrehozása, használata.
11.	Struktúrák, struktúra tömbök. Típusdefiníció.	Önhivatkozó struktúrák.	Struktúrák használata.
12.	Kétdimenziós tömbök, mutató tömbök.	Kétdimenziós tömbök algoritmusai, dinamikus memóriahasználat.	**2. programozás számonkérés (függvényírás, pointer)**
13.	A main függvény paraméterei és visszatérési értéke. Változó hosszúságú paraméterlistás függvények. A C fordító működése, az előfeldolgozó szerepe; előfordítónak szóló direktívák. A C99/C11 szabvány új elemei. Kódolási szabvány.	Fájlkezelés.	Struktúratömb, dinamikus memóriakezelés, fájlkezelés.
14.	**Elővizsga (írásbeli)**	**Elővizsga (programozás)**	**Programozás számonkérések pótlása**

Aki a 2. oktatási hét végéig nem írja alá a Hallgatói nyilatkozatot, miszerint megismerte és elfogadja a laborhasználati szabályzatot, oktatói felügyelettel sem tartózkodhat a tanszéki laborokban.

Tananyag:
www.iit.uni-miskolc.hu → Munkatársak: Baksáné V.E. → Oktatott tárgyak → Programozás alapjai

Ajánlott irodalom:

- Brian W. Kernighan, Dennis M. Ritchie: A C programozási nyelv, Az ANSI szerint szabványosított változat. Műszaki Könyvkiadó, Budapest, 1996.
- C.L. Tondo, S.E. Gimpel: C programozási gyakorlatok, Megoldások Brian W. Kernighan és Dennis M. Ritchie A C programozási nyelv című könyvének feladataihoz. Műszaki Könyvkiadó, Budapest, 1985.

Az aláírás megszerzésének feltételei:

1. Aktív részvétel a gyakorlatok 70%-án.
2. A 2 programozás számonkérés sikeres megoldása.

A vizsga menete: elmélet írásban és gyakorlat (programozás számítógépen)

Programozás alapjai
1. programozás számonkérés mintafeladat

A feladatot CodeBlocks fejlesztőkörnyezetben kell megoldani C programozási nyelven 30 perc alatt. Segítséget nem lehet igénybe venni a feladat megoldása során.

A feladat akkor elfogadható, ha:
- hiba nélkül lefordul (warning sem lehet) és futásidejű hiba nélkül lefut,
- az elvárt eredményt állítja elő,
- a feladathoz illeszkedő algoritmust helyesen alkalmazza,
- nem használ globális változót,
- a kód tabulált, jól olvasható; a változónevek beszédesek,
- a kód megfelel a feladatleírásnak,
- ellenőrzött adatbeolvasásnál a beolvasás sikerességét és a beolvasott érték helyességét is ellenőrzi.

Feladat:
Írjon C programot, amely megfelelő méretű egydimenziós tömbben nyilvántartja a benzin heti átlagárát egy negyedéven keresztül. Az első 2 hónap adatait inicializálja, majd implementálja az alábbi részfeladatokat:
1) A harmadik hónap adatait olvassa be ellenőrzött módon. A benzin ára 350-420 Ft között legyen.
2) Állapítsa meg és írja ki, hogy a benzin ára monoton csökkent-e az adott időszakban.
3) Írja ki a benzinárakat olyan sorozatként, ahol az első érték abszolút értékben adott, a többi az előzőhöz képest számított relatív érték.

Tesztadatok:

388.9
390.3
389.5

Eredmény:

A benzin ára monoton csökkent: nem
Benzinárak sorozata: 388.9 ; +1.4 ; -0.8

Programozás alapjai
2. programozás számonkérés mintafeladat

A feladatot CodeBlocks fejlesztőkörnyezetben kell megoldani C programozási nyelven 30 perc alatt. Segítséget nem lehet igénybe venni a feladat megoldása során.

A feladat akkor elfogadható, ha:
- hiba nélkül lefordul (warning sem lehet) és futásidejű hiba nélkül lefut,
- az elvárt eredményt állítja elő,
- a feladathoz illeszkedő algoritmust helyesen alkalmazza,
- nem használ globális változót,
- a kód tabulált, jól olvasható; a változónevek beszédesek,
- a kód megfelel a feladatleírásnak,
- ellenőrzött adatbeolvasásnál a beolvasás sikerességét és a beolvasott érték helyességét is ellenőrzi.

Feladat:
Írjon C programot saját függvények és a top-down programfejlesztési technika alkalmazásával. Keresse meg 10 és N között az összes relatív prím számpárt. N 10-nél nagyobb egész szám, amit ellenőrzötten olvasson be. A megtalált számpárokat írja ki a képernyőre.
Két szám egymáshoz képest relatív prím, ha legnagyobb közös osztójuk 1. A feladat megoldása során használja fel az euklidészi LNKO algoritmust:

Be: a, b
Amíg b>0
    segéd ← b
    b ← a/b maradéka
    a ← segéd
Ciklus vége
lnko ← a
Ki: lnko

Tesztadatok:
N = 15

Eredmény:
(10,11) ; (10,13) ; (11,12) ; (11,13) ; (11,14) ; (11,15) ; (12,13) ; (13,14) ; (13,15); (14,15)

Programozás alapjai
Írásbeli vizsga – minta

Beugró teszt (15 perc) - Minden helyes válasz 1 pontot ér. Minimum elvárás: 7 pont

1. Definiálja az operátorok kiértékelésének „asszociativitási szabályát”.
Az azonos precedencia szinten lévő operátorok végrehajtási sorrendjét határozza meg.

2. Írja le egy olyan függvény deklarációját, amely „Megkapja tömbben egy negyedévre a heti EURO árfolyamokat (valós számok) és visszaadja, hogy hányadik héten volt a legmagasabb az árfolyam”.
*int arfolyamMaxindex(double *arfolyamtomb, int tombmeret);*

3. Mi a *scanf* függvény visszatérési értéke?
a) A scanf függvénynek nincs visszatérési értéke (void típusú függvény).
b) A scanf függvény egy int-el tér vissza: 0 ha sikeres volt a beolvasás, egyébként sikertelen.
c) A scanf függvény egy int-el tér vissza: a sikeresen beolvasott adatértékek számával.

4. Melyik hivatkozás adja meg a tömb első elemének az értékét?
a) tomb[1]
b) tomb[0]
c) *tomb[0]

5. Mit határoz meg a változó típusa?
a) a memóriaszegmenst, ahol a változó létrejön
b) a változó hatáskörét
c) a memóriában lefoglalt terület méretét

6. Definiálja az élettartam fogalmát.
A változó élettartama a program végrehajtásának azon szakasza, amely alatt a változó a memóriában helyet foglal.

7. Mi lesz x értéke?
a) 2
b) 5
c) Hibás kifejezés, a fordító nem tudja értelmezni.

```
int a=5, b=2;
int x = a > b ? a : b;
```

8. Írja át a for ciklust while ciklusra.

```
int i;
for (i=10; i>0; i--) {
        printf("%d ", i);
}
```

```
int i=10;
while (i>0) {
        printf("%d ", i);
        i--;
}
```

9. Definiálja a változó fogalmát.
A változó egy azonosítóval elnevezett memória terület. Jellemzői: a neve, memóriacíme, típusa, pillanatnyi értéke.

10. Definiálja az algoritmus fogalmát.
Egy feladat megoldásának véges számú lépésben történő egyértelmű és teljes leírása.

Programozás alapjai
Írásbeli vizsga – minta

Az írásbeli vizsga folytatásának feltétele a beugró teszt sikeres teljesítése. Az írásbeli vizsga szintenként nehezedő elméleti kérdésekből áll. 60 perc áll rendelkezésre, a megoldás során semmilyen segédeszköz nem használható.

2-es szint:

1. Melyek a C nyelv szelekciós utasításai? Adja meg a szintaktikájukat, jellemezze működésüket! 10 pont

Szelekció: utasítások feltételtől függő végrehajtása
Szelekciós utasítások: if – else ; if – else if – else ; switch – case

if (kifejezés) { *utasítás1* *} else {* *utasítás2* *}*	*Ha a „kifejezés” igaz (értéke nem nulla), „utasítás1” kerül végrehajtásra, egyébként (ha „kifejezés” értéke nulla) „utasítás2”.*
if (kifejezés1) { *utasítás1* *} else if (kifejezés2) {* *utasítás2* *} else {* *utasítás3* *}*	*A program többirányú elágaztatását teszi lehetővé. Ha „kifejezés1” igaz (értéke nem nulla), „utasítás1” kerül végrehajtásra, egyébként kiértékelődik „kifejezés2”. Ha ez igaz (értéke nem nulla), „utasítás2” hajtódik végre. Egyébként (ha „kifejezés1” és „kifejezés2” értéke is nulla), akkor „utasítás3” lesz végrehajtva. Az „else if” ágak megengedett száma implementáció függő.*
switch(kifejezés) { *case konst1: utasítás1* *case konst2: utasítás2* *…* *default: utasítás* *}*	*A switch kifejezés csak egész típusú lehet (int vagy char). Kiértékelődik a kifejezés és a végrehajtás arra a case címkére ugrik, amelyik konstans megegyezik a kifejezés értékével. Ettől a ponttól kezdve a blokkon belül minden utasítás végrehajtásra kerül. Ezért az esetek szétválasztásához az egyes esetekhez tartozó utasításokat a break megszakító utasítással zárjuk. A default címke opcionális. Szerepe: ha a kifejezés egyik konstanssal sem egyenlő, a default címkéhez tartozó utasítás hajtódik végre.*

2. Mi a tömb definíciója? Hogyan tároljuk és kezeljük a tömböket C-ben? 10 pont

Tömb: azonos típusú adatok tárolására szolgáló, nem bővíthető (fix méretű) összetett adatszerkezet.
Tömbök tárolása C-ben:
A C fordító a tömbben tárolt adatokat a memóriában egy adott kezdőcímtől kezdve (sor)folytonosan tárolja. Ezt a kezdőcímet a tömbváltozó tárolja.
Tömbök kezelése C-ben:
Tömbökre semmilyen művelet nincs definiálva. A tömböt elemenként lehet kezelni. Az egyes elemek elérése az indexükkel lehetséges a tömbindexelés operátorával (pl. tomb[0]). A tömb indexelése 0-tól méret-1-ig tart. Az indextartomány túllépését a C fordító nem ellenőrzi.

3. Írja le a függvény fogalmát! Függvényhíváskor hogyan történik C-ben az argumentumok átadása a hívott függvénynek, hogyan kezeli a fordító ezeket az adatokat a hívott függvényen belül és mi ennek a következménye? Mit tud a függvény visszatérési értékéről? 10 pont

Függvény: a program olyan névvel ellátott alprogramja, amely a program bármely részéből annyiszor hívható, ahányszor szükség van a függvényben definiált tevékenység sorozat végrehajtására.
Argumentumok átadása: az argumentumok érték szerint adódnak át a függvény paramétereinek. A fordító a fv híváskor megadott argumentumlista és a fv paraméterlistája közötti egyezést vizsgálja: azok száma és típusa rendre meg kell egyezzen.
A fv paraméterei az átadott értékek másolatát veszik fel kezdőértékként. A paraméterek a fv lokális változói: élettartamuk a fv futásának idejére terjed ki, csak a fv-ben láthatók.
Következmény: a fv futása során a paraméterek értékében bekövetkezett változások a fv-ből való visszatérés után nem láthatók.
Visszatérési érték: fv visszatérési értéke lehet bármilyen skalár vagy struktúra típusú. A visszatérési érték a return utasításban megadott kifejezés értéke. Típusa a fv visszatérési típusa. Az olyan fv-ek, amelyek nem adnak vissza értéket, void visszatérési típusúak.

3-as szint:

4. Definiáljon emberek adatainak nyilvántartására alkalmas struktúra típust (név, születési dátum : év, hónap, nap) háromféleképpen: egyetlen struktúrával, struktúra beágyazással és külön definiált dátum típus felhasználásával. 10 pont

```
typedef struct person {
    char name[20];
    int year, month, day;
} Person;
```

```
typedef struct person {
    char name[20];
    struct {
        int year, month, day;
    } birthdate;
} Person;
```

```
typedef struct date {
    int year, month, day;
} Date;
typedef struct person {
    char name[20];
    Date birthdate;
}
```

4-es szint:

5. Mit határoz meg egy változó típusa, és mit a tárolási osztálya? Melyek a C alapértelmezett tárolási osztályai és ezek milyen élettartamot, láthatóságot biztosítanak? 10 pont

A változó típusa meghatározza a tárolási méretet és az elvégezhető műveletek körét.
A tárolási osztály meghatározza hol jön létre a változó a memóriában és a változó élettartamát.
Alapértelmezett tárolási osztályok: függvények és globális változók esetén „extern”, lokális változók esetén „auto”.
extern TO: globális élettartam, programszintű láthatóság
auto TO: lokális élettartam, blokk szintű láthatóság

5-ös szint:

6. Sztringek tömbjének tárolására milyen megoldásokat ismer C nyelvben? Írjon példát, melyiket milyen esetben célszerű használni. Hogyan történik az így tárolt sztringek kezelése? 10 pont

- Kétdimenziós tömb alakban: char strArray[num][length]
*Ahol num a tömbben tárolt sztringek száma, length pedig a sztringek hossza. Az egyes dimenziókban az indexelés 0-tól méret-1-ig tart. A sztringet lezáró '\0' karaktert a sztring hosszánál kell figyelembe venni. A definíció során num*length méretű karaktertömb jön létre; a fordító ekkora területet foglal a memóriában folytonosan. Akkor használjuk, amikor azonos hosszúságúak a sztringek (pl. hónapnevek 3 betűs rövidítése). Csak az első index megadásával az adott sztringre hivatkozunk. Mindkét index megadásával az első index szerinti sztring adott karakterére hivatkozunk.*

*- Mutatótömb alakban: char *strArray[num]*
Mutatótömb: mutatókat tároló tömb. Olyan vektor (egydimenziós tömb), amelynek elemei vektorok kezdőcímét kijelölő mutatók. A sztringek hatékonyabb tárolását teszi lehetővé. Akkor használjuk, ha a tömbben tárolt sztringek különböző hosszúságúak. Kezelése uaz mint amikor 2D tömbként tároljuk.

Értékelés:
A dolgozatot szintenként javítjuk. Egy szintet akkor sikerült teljesíteni, ha a hallgató elérte az összpontszám 70%-át. A következő szintet akkor javítjuk, ha az előzőek sikerültek. Azaz,
- a 2-es szint teljesítéséhez az első 3 feladatból minimum 21 pontot kell elérni.
- a 3-as szint teljesítéséhez a 2-es szint teljesítése után a 4. feladatból minimum 7 pontot kell elérni.
- a 4-es szint teljesítéséhez a 2-es és 3-as szint teljesítése után az 5. feladatból minimum 7 pontot kell elérni.
- az 5-ös szint teljesítéséhez a 2-es, a 3-as és a 4-es szint teljesítése után a 6. feladatból minimum 7 pontot kell elérni.

Programozás alapjai
Programozás vizsga – minta

Az írásbeli vizsga sikeres teljesítése után a vizsga programozással folytatódik. A hallgatók az írásbeli vizsgán elért szintnek megfelelő programozási feladatot kapnak. Ha ezt sikerül megoldaniuk, megvédték az írásbeli eredményét. Amennyiben nem sikerül megvédeni az írásbeli eredményét, a vizsgajegy meghatározásakor a feladatmegoldás tényleges szintjét vesszük figyelembe (lásd „Vizsgán számonkért programozási ismeretek").

A feladatot CodeBlocks fejlesztőkörnyezetben kell megoldani C programozási nyelven 30 perc alatt.
Segítséget nem lehet igénybe venni a feladat megoldása során.

A feladat sikeres, ha:
- hiba nélkül lefordul (warning sem lehet) és futásidejű hiba nélkül lefut,
- az elvárt eredményt állítja elő,
- a feladathoz illeszkedő algoritmust helyesen alkalmazza,
- nem használ globális változót,
- a kód tabulált, jól olvasható; a változónevek beszédesek,
- a kód megfelel a feladatleírásnak,
- ellenőrzött adatbeolvasásnál a beolvasás sikerességét és a beolvasott érték helyességét is ellenőrzi.

2-es szintű feladat:
Írjon C programot, amelyben deklarál egy 10 elemű integer tömböt. Írjon külön függvényeket az alábbi feladatok megoldására:
a) Töltse fel a tömböt 1 és 10 közé eső véletlenszámokkal. Használja az *stdlib.h srand()* és *rand()* függvényeit.
Véletlenszámgenerátor mag inicializálása: *srand(time(0));*
(A *time()* fv deklarációja a *time.h*-ban található)
Véletlenszám előállítása: *rand()%(felsőhatár-alsóhatár+1)+alsóhatár*
b) Írja ki a tömb elemeit növekvően rendezve (minimum kiválasztásos rendezéssel).
c) Állapítsa meg, hogy a tömbelemek között van-e ismétlődő szám és számolja meg hányszor fordul elő a sorozatban. Pl.: a rendezett sor: 1, 3, 4, 4, 5, 7, 8, 8, 8, 10. A 4-es szám 2-szer fordul elő, a 8-as szám 3-szor fordul elő.

3-as szintű feladat:
Készítse el a négy alapművelet végrehajtását megvalósító C nyelvű kalkulátor programot. Ne használja az if utasítást! A felhasználói input alakja kétféle lehet (mindkettőt kezelje a program):
- egy egész szám, egy operátor és mégegy egész szám, pl. 12 + 5. Az egész szám beolvasását ellenőrzött módon végezze a program. Az elfogadott operátorok: +, –, *, /
- egyetlen sztring, pl. "12*4". Ebben az esetben az *sscanf(str, formátumleírás, tárolási címek)* függvénnyel tudja kiolvasni az input sztringből a komponenseket. Ellenőrizze, hogy sikeres volt-e a beolvasás (a függvény visszatérési értéke ugyanaz mint a *scanf* függvényé). Az elfogadott operátorok: +, –, *, /
Az eredmény kiszámítását külön függvényekben valósítsa meg a műveleti jeltől függően.

4-es szintű feladat:

Készítsen másodfokú egyenlet megoldó C programot. Külön függvényben olvassa be a másodfokú egyenlet konstansait. A felhasználói input alakja kétféle lehet (mindkettőt kezelje a program):

- három egész szám. Az egész szám beolvasását ellenőrzött módon végezze a program.
- egyetlen sztring, pl. ”5, -3, -2”. Ebben az esetben az *sscanf(str, formátumleírás, tárolási címek)* függvénnyel tudja kiolvasni az input sztringből a komponenseket. Ellenőrizze, hogy sikeres volt-e a beolvasás (a függvény visszatérési értéke ugyanaz mint a *scanf* függvényé).

Írjon függvényt, amely kiszámítja és visszaadja a másodfokú egyenlet 2 valós gyökét egy struktúrában.

Másodfokú egyenlet: $ax^2 + bx + c = 0$ (ahol a nem 0)

Diszkriminánsa: $D = b^2 – 4ac$

Ha D>0 két különböző valós gyöke van, ha D=0 egy valós gyöke van (a két valós gyök értéke megegyezik).

Megoldóképlet:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

5-ös szintű feladat:

Az egyetem alkalmazottairól a következő adatokat szeretnénk tárolni megfelelő adatszerkezetben: név, kar, munkaviszony kezdete (év), fizetés. A karokat {GÉIK, MFK, MAK, ÁJK, GTK, BTK, EGK, BZI} a felsorolt adattípus használatával adja meg. Írjon C programot, amely az alábbi műveleteket külön függvényben valósítja meg:

1) Beolvassa egy alkalmazott adatait, és adatbevitelkor ellenőrzi, hogy a fizetés ne legyen alacsonyabb a minimálbérnél (2018-ban 138 000 forint).

2) Ha a GÉIK kar alkalmazottja, a fizetését 10%-al növeljük.

3) Kiírja az alkalmazott adatait egy olyan szövegfájlba, amelyhez minden programfutáskor hozzáfűzzük az újonnan felvitt alkalmazott adatait.

Programozás alapjai
Vizsgán számonkért elméleti témakörök

2-es szint:

- Algoritmus és leírási módszerei.
- Adat, adatszerkezet, adattípus, változó, konstans fogalmai.
- Láthatóság és élettartam fogalmai.
- Standard I/O könyvtári függvények, standard header állományok.
- Kifejezések és operátorok. Operátorok precedenciája, kifejezések kiértékelése.
- Implicit, explicit típuskonverzió.
- A C nyelv utasításai. Vezérlési szerkezetek C nyelvi megvalósítása.
- Egydimenziós numerikus tömbök tárolása, kezelése.
- Függvény definiálása, deklarálása. Függvényhívás mechanizmusa, paraméterátadás.
- Alapalgoritmusok: számlálás, összegzés, eldöntés, kiválasztás, keresés, rendezés minimum/ maximum kiválasztással.

3-as szint: az eddigiek és

- Struktúra definiálása, típusdefiníció.
- Véletlenszám generátor működése.
- Egyszeres indirektségű mutatók (pointer).
- Sztring fogalma, tárolása és kezelése.
- Programtervezési alapelvek.
- Rendezések.

4-es szint: az eddigiek és

- A main függvény paraméterei és visszatérési értéke.
- A C fordító működése. Az előfeldolgozó szerepe; előfordítónak szóló direktívák.
- A C fordító memóriakezelése. Dinamikus memóriahasználat.
- Tárolási osztályok. Változók és függvények élettartama, láthatósága.
- Enumerációs adattípus.
- Makroeljárások definiálása, használata.

5-ös szint: az eddigiek és

- Változó hosszúságú paraméterlistás függvények.
- Kétdimenziós tömbök, mutató tömbök.
- Rekurzió.
- Moduláris programozás. Saját header állomány készítése.
- Fájlkezelés.
- Önhivatkozó struktúrák.

Programozás alapjai
Vizsgán számonkért programozási ismeretek

2-es szint:

- Konstansok és változók deklarálása, inicializálása.
- Elemi adattípusok használata.
- Standard I/O könyvtári függvények hívása, standard header állományok beillesztése.
- Standard matematikai rutinok hívása (math.h).
- Kifejezések és operátorok. Operátorok precedenciája, kifejezések kiértékelése.
- A C nyelv utasításai. Vezérlési szerkezetek.
- Egydimenziós numerikus tömbök használata. Tömbi algoritmusok.
- Saját függvény írása, meghívása.
- Alapalgoritmusok: számlálás, összegzés, eldöntés, kiválasztás, keresés, rendezés minimum/ maximum kiválasztással.
- Implicit, explicit típuskonverzió.

3-as szint: az eddigiek és

- Struktúra típus definiálása, típusdefiníció. Struktúra változó deklarálása, használata.
- Makroszimbólumok használata.
- Véletlenszám generátor használata tetszőleges intervallum esetén.
- Egyszeres indirektségű mutatók használata.
- Sztringek kezelése. Standard sztringkezelő függvények (string.h) használata.

4-es szint: az eddigiek és

- Dinamikus memóriakezelés.
- Struktúra mint függvény visszatérési érték és mint függvény argumentum.
- Struktúra pointer deklarálása. Struktúra pointer mint függvény visszatérési érték és mint függvény argumentum.
- Makroeljárások készítése, hívása.
- Enumerációs adattípus használata.

5-ös szint: az eddigiek és

- Kétdimenziós tömbök, sztringtömbök megadása és argumentumként történő átadása.
- Struktúra tömb definiálása, feltöltése, kezelése.
- Több modulos program, saját header állomány készítése.
- Rekurzív függvények használata, rekurzív és iteratív algoritmusok közötti konverzió.
- Fájlkezelés.