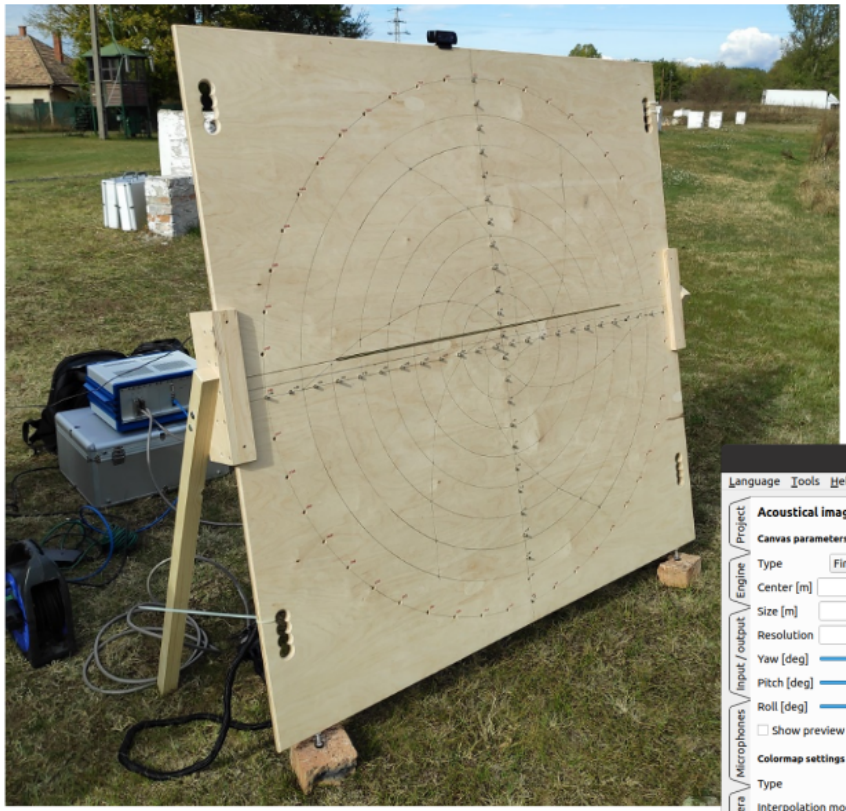
A programozás alapjainak oktatása a BME Villamosmérnöki és Informatikai Karán

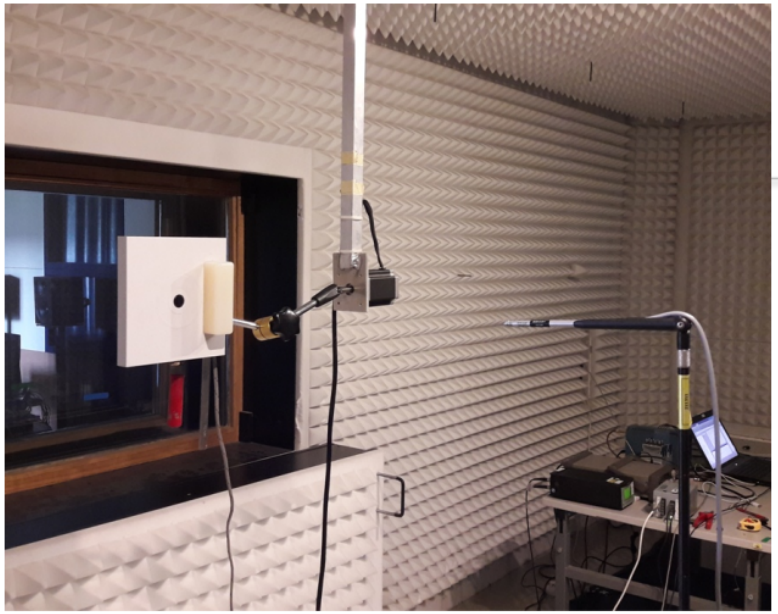
Fiala Péter, Imre Sándor

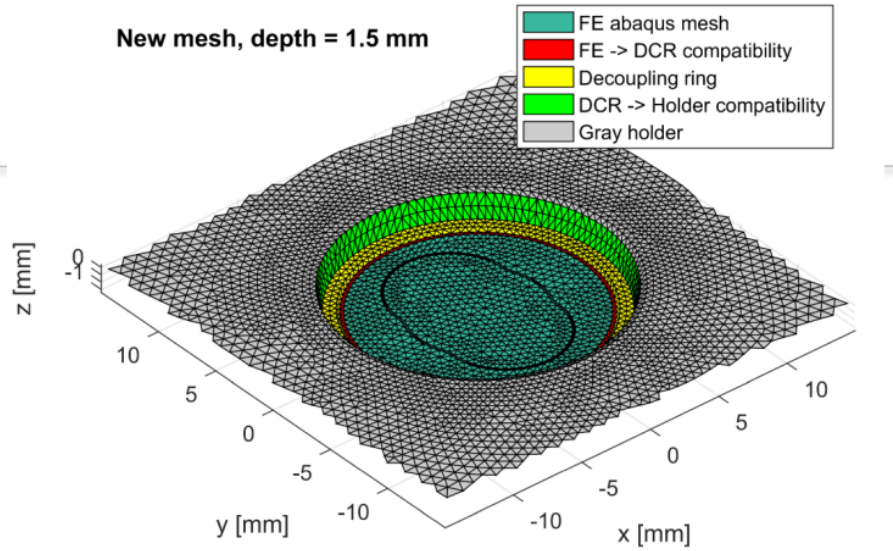
INFO ÉRA 2025
2025. április 25.

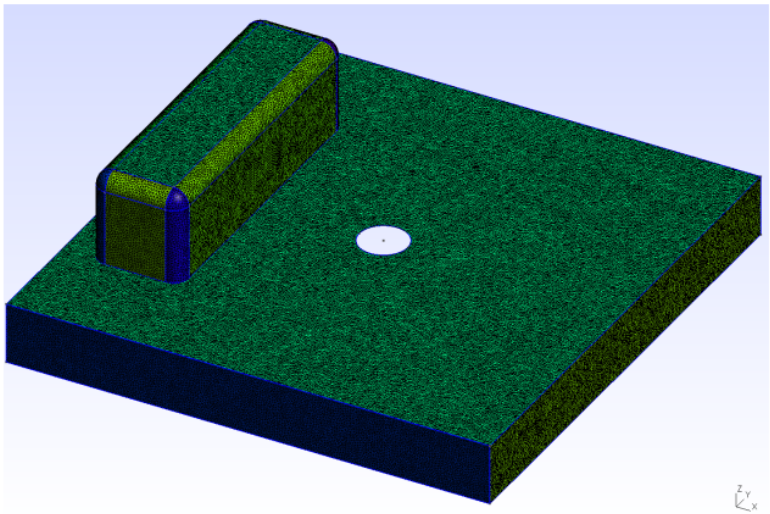
Bemutatkozás – programozó villamosmérnök

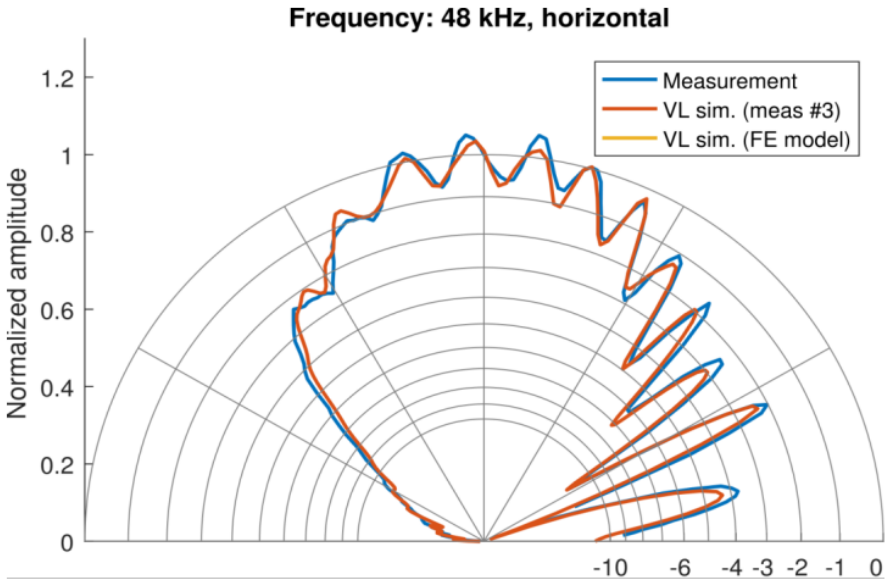


Bemutatkozás – programozó villamosmérnök









Tartalom

Kiket oktatunk? – BSc képzés

szak	létszám	ponthatár	programozott
Villamosmérnöki	300	320	30%
Mérnökinformatikus	700	340	80%
Üzemmérnök-informatikus	50	321	n.a. %

- Közös felvételi követelmény: két pontszerző érettségi tárgy közül
  - egyik a matematika
  - egyik emelt szintű

- Nagy szakokon 70% emelt szintű matematika érettségi
- Középiskolából hozott programozási nyelv
  - C#
  - Python
  - (C++)

Programozásoktatás a BSc Villamosmérnöki Szakán

Informatika II – SQL, JavaScript, PHP, TypeScript

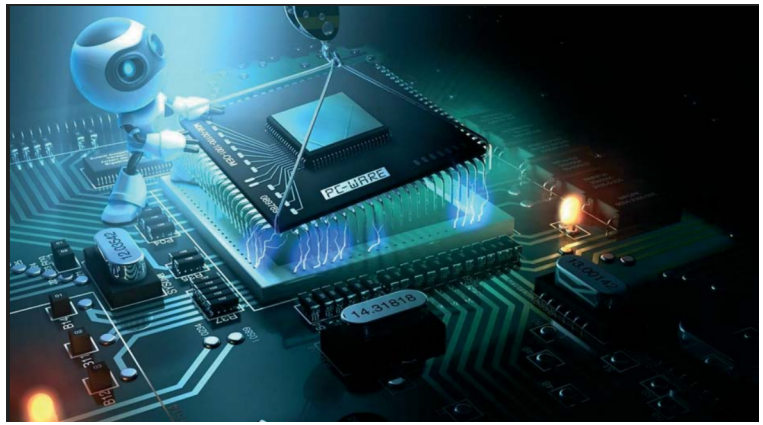
Laboratórium – VeriLog, VHDL

Szabályozástechnika – Matlab

A programozás alapjai II – C++

Digitális technika II – Assembly

A programozás alapjai I – C



Programozásoktatás a BSc Mérnökinformatikus Szakán

Szoftvertechnika – C#, .NET

Projekt labor

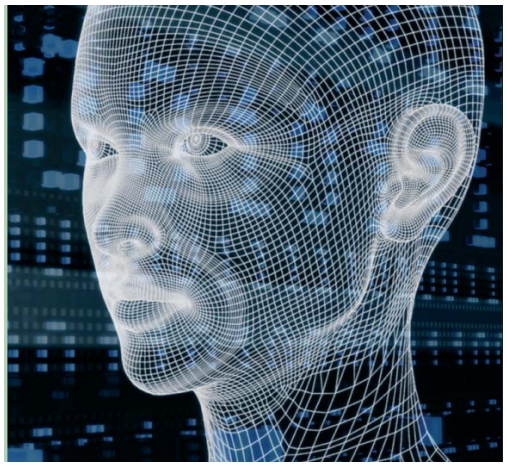
Grafika – OpenGL, GLSL, CUDA

A programozás alapjai III – Java

Adatbázisok – SQL

Szoftvertechnológia – UML

A programozás alapjai II – C++



A programozás alapjai I – C

Digitális technika – VeriLog, Assembly

Programozásoktatás az Üzemmérnök-Informatikus Szakon

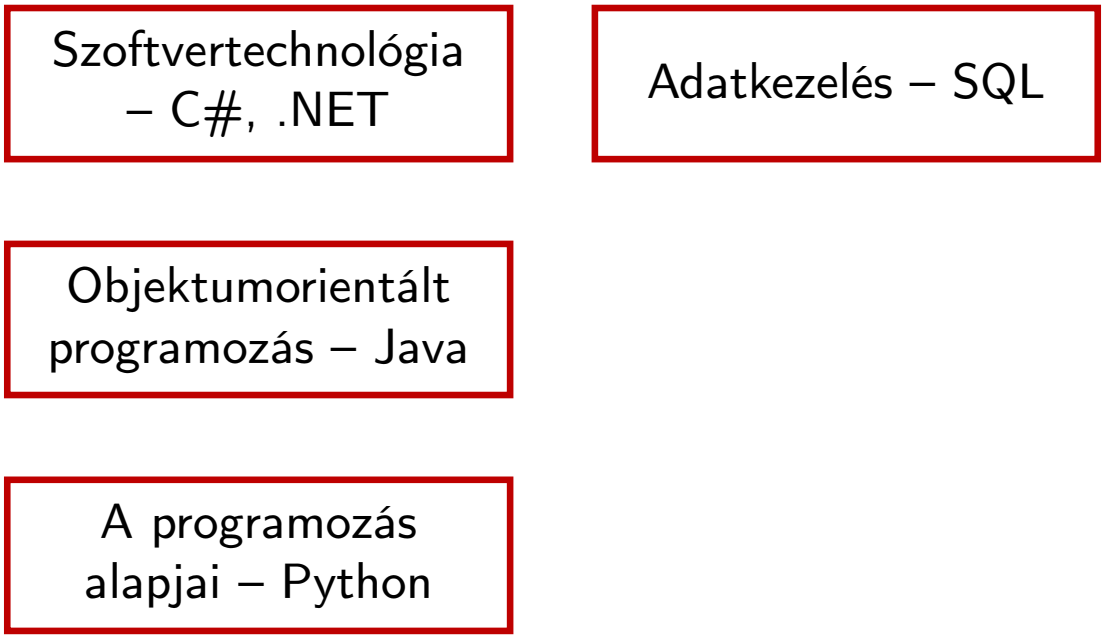
Szoftvertechnológia – C#, .NET

Adatkezelés – SQL

Objektumorientált programozás – Java

A programozás alapjai – Python

Programozásoktatás az Üzemmérnök-Informatikus Szakon



- Szabadon választható tárgyak (70 db) keretein belül
  - iOS, Android, Kotlin, Flutter, Matlab, Python, LUA, JavaScript, C++20, ...

A programozás alapjai – Miért C és C++?

- kontra
  - 50 éves!
  - Nem tanulónyelv
  - Nincs sztring!
  - Tömör szintaxis
  - Könnyű hibázni, nehéz megtalálni
- pro
  - Kis nyelv
  - Direkt nyelv (de, tanulónyelv!)
  - Fordított nyelv
  - Hardver- és operációsrendszer-közeli
  - Hordozható
  - C++ absztrakt ÉS hatékony
  - Kivételesen jól optimalizálható

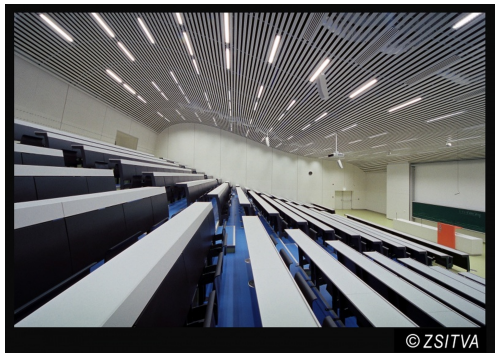
```
10
11  while (*to++ = *from++);
12
```

```
13  char c;
14  do {
15      c = from[i];
16      to[i] = c;
17      i++;
18  }
19  while (c != '\0');
```

```
20
21  map<string,int> stats;
22  string word;
23  while (cin >> word)
24      stats[word]++;
25
```

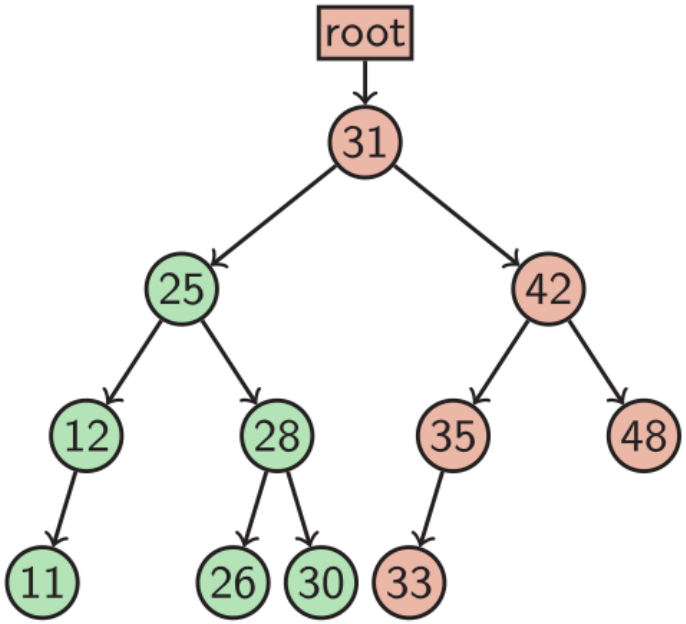
A programozás alapjai 1 (infó és villamos)

- **2/2/2/f/7**, a képzés legnagyobb kiméretű tantárgyai



1. Előadás
   - Elmélet, érdeklődés és motiváció felkeltése nagyelőadókban
2. Táblás gyakorlat
   - Részletkérdések, vezetett gyakorlás 30 fős tankörökneknek
3. Számítógépes labor
   - Önálló feladatmegoldás oktatói segítséggel 20 fős csoportoknak
4. Otthoni önálló munka
   - A befektetett munka oroszlánrésze

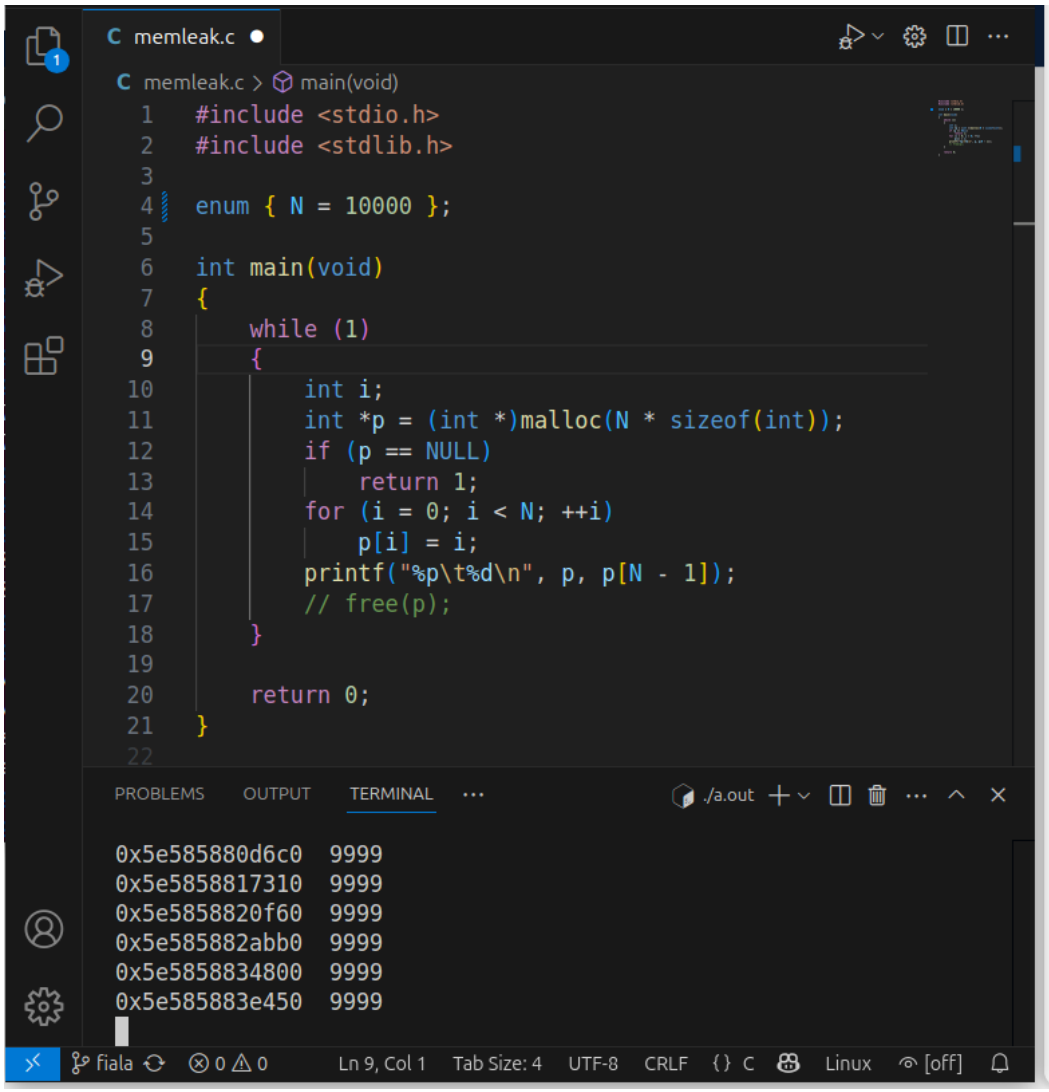
Bejárás – preorder

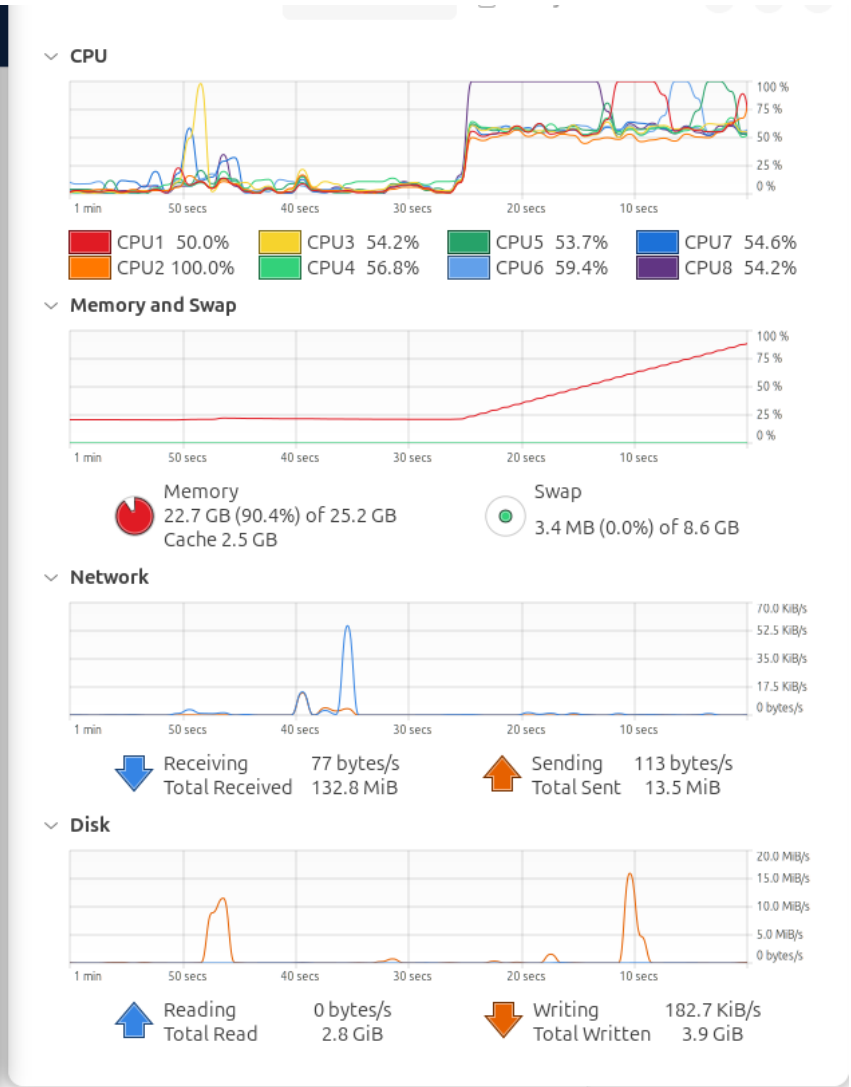



```
1  void preorder(tree_ptr root)
2  {
3    if (root == NULL)
4      return;
5    printf("%d ", root->data);
6    preorder(root->left);
7    preorder(root->right);
8  }
```

```
31 25 12 11 28 26 30
```

Nagytermi előadások





Kistermi táblás gyakorlatok

INFOC INFÓ ▸ TANANYAG ▸ SEGÉDLET ▸ EXTRÁK ▸ BELÉPÉS

1. Negyedik kis ZH

Ezen az órán van a negyedik kis ZH.

2. Gyors hatványozás

A hatványozás elvégezhető annál gyorsabban is, mintha a kitevőnek megfelelő számú szorzást csinálnánk. Pl. $a^8 = a^4 \cdot a^4$, továbbá $a^4 = a^2 \cdot a^2$, és végül $a^2 = a \cdot a$ miatt a nyolcadikra hatványozáshoz mindössze három szorzásra van szükség. A következő megfigyelést tehetjük:

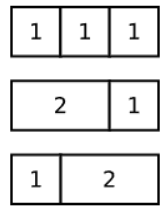
- $a^k = a^{k/2} \cdot a^{k/2} = (a \cdot a)^{k/2}$, ha k páros, és
- $a^k = a \cdot a^{k-1}$, ha k páratlan.

Írjunk rekurzív függvényt, amely a fentiek alapján végzi el a hatványozást! Paraméterei legyenek az alap és a kitevő, visszatérési értéke pedig a keresett hatvány!

▸ Megoldás

3. Járda kövezése

Hányféleképpen lehet egy adott hosszúságú járdát kikövezni 1 és 2 méter hosszúságú járdalapokkal? Például ha 3 méteres a járda, a lehetőségek: 1+1+1, 1+2, 2+1, tehát összesen 3.



▸ Megoldás

Hogyan kell módosítani a programot, ha 3 egység hosszú lapunk is van?

Hogyan, ha paraméterként kapjuk, hogy hányféle lap van? (Pl. 5 → 1, 2, 3, 4, 5 hosszú lapok vannak.)

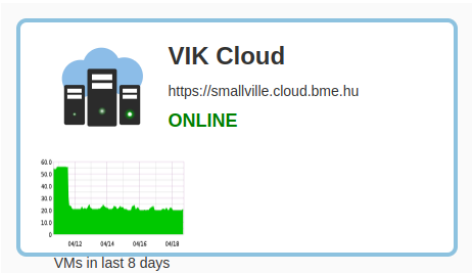
▸ Megoldás

4. Permutáció

Írjuk ki egy egész tömbnek (számsornak) az összes permutációját, azaz lehetséges sorrendezését! Pl. 123 permutációi: 123, 132, 213, 231, 312, 321.

Számítógépes laborok

moodle



VMs in last 8 days

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
Faculty of Electrical Engineering and Informatics
MŰEGYETEM 1782

Home / My courses / A programozás alapjai 1 - BMEVIHIAA01 2024/25/1 / Feladatbank / Pozitívak visszafelé

Navigation

- Home
  - Dashboard
  - Site pages
  - My courses
    - A programozás alapjai 2 - BMEVIIIAA03 2024/25/2
    - Akusztika - BMEVIHIMA19 2024/25/2
    - Akusztika és hangtechnika laboratórium - BMEVIHIMB...
    - Hangszerek fizikája - BMEVIHIJV68 2024/25/2
    - Kutatás 7. - BMEVIHIDK07 2024/25/2
    - Publikáció 7. - BMEVIHIDP07 2024/25/2
    - A programozás alapjai 1 - BMEVIHIAA01 2018/19/1

Pozitívak visszafelé

Beküldési határidő: 2024. 10. 01. 15 óra 00 perc (UTC)

Írj programot, mely egész értéket olvas a bemenetről, és ezek közül a pozitívokat fordított sorrendben kiírja! Az adatsor végjele a 0 érték, legfeljebb 100 darab érték érkezhet (a végjel előtt).

Például: `2 -1 5 -7 -1 8 -3 0 -> 8 5 2`

A határidő lejárt, így a további próbálkozásokkal nem szerezhető pont.

Megoldás beküldése

Otthoni önálló munka

INFOC INFÓ TANANYAG SEGÉDLET EXTRÁK BELÉPÉS

11. Közvetlen kiválasztással (selection sort)

1 2 3 4 5 6 7 9 8

legkisebb: 5...

start következő folyamatos

Lényege: megkeresi a rendezetlen tömbrészlet legkisebb elemét, és az elejére rakja.

Ezt az algoritmust szélsőértékkeresős, vagy minimumkeresős rendezésnek is szokták nevezni. A működéséhez a buborék algoritmusnál tett megfigyelés adja az ötletet: ott a belső ciklus minden futása után a legnagyobb elem a rendezetlen részlet végére, és ezáltal a rendezett részlet elejére került. Az ötlet lényege, hogy ne cserékkel toljuk el odáig a legnagyobb elemet, hanem inkább keressük meg a tömbben azt, és végezzük el egy lépésben a cserét. Vagyis tegyük egyből a helyére a kérdéses elemet. Itt a legkisebb elemekkel történik ez.

A közvetlen kiválasztásos algoritmus előnye a buborékrendezéshez képest, hogy jóval kevesebb cserét végez a tömbben. Itt mindegyik tömbelem egy lépésben a helyére kerül, vagyis legrosszabb esetben is a cserék száma db-1, ahol db a tömb mérete.

```
void kozvetlen(double *t, int db) {
    for (int i = 0; i < db-1; ++i) {
        int minindex = i;                    // minimum keresése
        for (int j = i+1; j < db; ++j)
            if (t[j] < t[minindex])
                minindex = j;

        if (minindex != i) {
            double temp = t[minindex];
            t[minindex] = t[i];              // csere.
            t[i] = temp;
        }
    }
```

infoC portál

Rekurzió. Fák.

Előadásfóliák

Fóliasor - handout

Fóliasor - slide

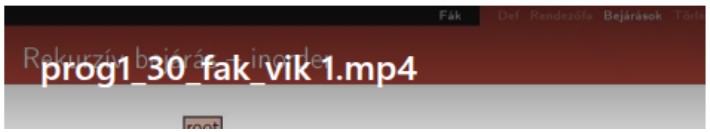
Videóleckék

Rekurzió

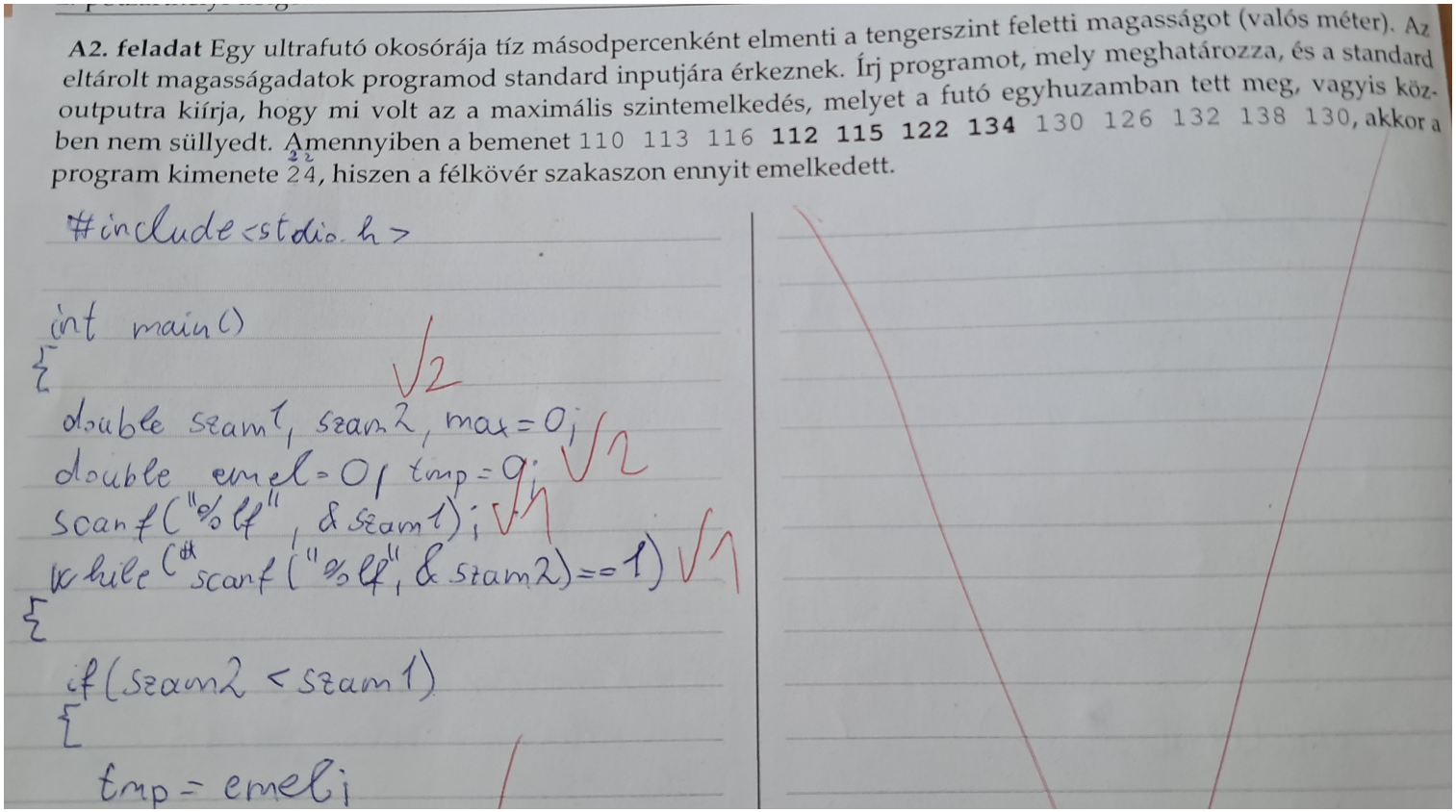
Ellenőrző kérdések

1. Miért nem hatékony a Fibonacci-sorozat rekurzív kiértékelése? Mit kellene tennünk a rekurzív kiértékelés meggyorsításához?

Fák



Számonkérések – 1. félévben csak papíron



A2. feladat Egy ultrafutó okosórája tíz másodpercenként elmenti a tengerszint feletti magasságot (valós méter). Az eltárolt magasságadatok programod standard inputjára érkeznek. Írj programot, mely meghatározza, és a standard outputra kiírja, hogy mi volt az a maximális szintemelkedés, melyet a futó egyhuzamban tett meg, vagyis közben nem süllyedt. Amennyiben a bemenet 110 113 116 **112 115 122 134** 130 126 132 138 130, akkor a program kimenete 22, hiszen a félkövér szakaszon ennyit emelkedett.

```
#include<stdio.h>

int main()
{
double szam1, szam2, max=0;
double emel=0, tmp=0;
scanf("%lf", &szam1);
while(scanf("%lf", &szam2)==1)
{
 if(szam2 < szam1)
 {
  tmp = emeli
```

- mert a hallgatónak sokkal könnyebb
- szintaxis nem sorsdöntő, nem kell, hogy forduljon
- **elvileg** helyes megoldásokat várunk
- 300 nagy zh-t estére javítunk

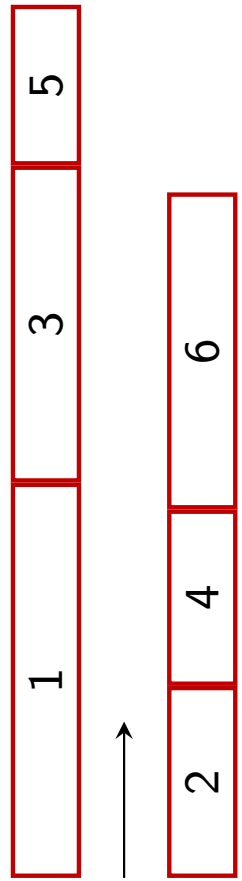
Kólagép – az elvárás a 7. héten

Az Informatika épület 32 rekeszes kólagépéből kólát, fantát, cappyt, burnt vagy bubis vizet lehet venni. Ha a vevő beüti egy rekesz sorszámát, akkor a kért italt kapja, de nem feltétlenül abból a rekeszből, amit beütött. Az automata ugyanis nem akarja leüríteni rekeszeit, ezért a kiválasztott italt abból a rekeszből adja ki, melyben abból a legtöbb van.

Válassz adatszerkezetet az automata ábrázolásához, és hozz létre egy automatát. Olvasd be egy rekesz sorszámát (kérés), és megfelelően módosítsd az automata állapotát.

Kossuth utca

A Kossuth utcában 1-től 100-ig minden házszámhoz tartozik telek. A páratlanok a bal, a párosak a jobb oldalon vannak. A postás minden reggel végigbiciklizik a Kossuth utca közepén, és amint egy telek sarkával egyvonalba kerül, behajítja a kertbe (jobbra vagy balra) a napilapot. Írj programot, mely házszám szerinti sorrendben beolvassa a Kossuth utca telkeinek szélességeit, majd kiírja, hogy milyen sorrendben hajítja be a postás az újságot az egyes kertekbe.



Amennyiben a program bemenete 25 12 20 12 10 20 ..., akkor kimeneten a házszámok elvárt sorrendje 1 2 4 6 3 ...

Számonkérések – 2. félévtől csak gépen

Question **1**

Partially correct

Marked out of 7.00

Flag question

Time left 0:15:05 Hide

Készítsen egy osztályt (*Ido*) óra (*ora*) és perc (*perc*) tárolásához! Az osztálynak legyen

- paraméter nélkül hívható konstruktora, ami 12 óra 22 percet állít be;
- olyan konstruktora is, amivel az óra és a perc is megadható (mindkettő egész) ;
- olyan tagfüggvénye (*show*) amivel oo:pp formátumban kiírathatók az objektum adatai a standard kimenetre;
- olyan operátora(+), amivel egy időponthoz egész (int) órát lehet hozzáadni jobbról! Az eredmény az összeadás műveletnél megszokott módon új objektumban keletkezzen! Amennyiben az összeadás során keletkező objektumban az óra értéke meghaladná a 23 órát, úgy dobjon *const char** kivételt, melynek értéke az Ön Neptun azonosítója legyen!

Javaslat: próbálja meg először a konstruktort (-torokat) implementálni és azt ellenőrizni! Sajnos, ha a konstruktor hibás/nincs, akkor nem tudjuk tesztelni az osztályt, így pontot sem kap!

Valósítsa meg az osztályt C++ nyelven! **Használja** a zárójelben levő neveket! **Ne lehessen** az adattagokhoz kívülről közvetlenül hozzáférni! **Működjön** az elvárásnak megfelelően az alábbi kódrészlet:

```
Ido t0, t10(10, 3), t23(23, 5);  // 12:22, 10:10, 23:05 időpontokat hoz létre
t0 = t10 + 1;                     // t0-ba 11:03 kerül, t10 változatlan marad
t0.show();                       // KIIR: 11:03
t23 + 2;                          // kivétel keletkezik
// konstans objektumra is működnie kell az osztálynak!
const Ido tc;
(tc + 1).show();
```

Answer: (penalty regime: 0 %)

Reset answer

```
1  #include <iostream>
2  #include <iomanip>
3
4  class Ido {
5      int ora, perc;
6  public:
7      Ido(int ora = 12, int perc = 22) : ora(ora), perc(perc) {}
8      void show() const { std::cout << ora << ":" << perc; }
9  };
```

Számonkérések – 2. félévtől csak gépen

Question **1**

Partially correct

Marked out of 7.00

Flag question

Time left 0:15:05 Hide

Készítsen egy osztályt (*Ido*) óra (*ora*) és perc (*perc*) tárolásához! Az osztálynak legyen

- paraméter nélkül hívható konstruktora, ami 12 óra 22 percet állít be;
- olyan konstruktora is, amivel az óra és a perc is megadható (mindkettő egész) ;
- olyan tagfüggvénye (*show*) amivel oo:pp formátumban kiírathatók az objektum adatai a standard kimenetre;
- olyan operátora(+), amivel egy időponthoz egész (int) órát lehet hozzáadni jobbról! Az eredmény az összeadás műveletnél
- n az óra értéke

Java
tudju
Való
hozz
Ido
t0 =
t0.s
t23
// k
cons
(tc

or hibás/nincs, akkor nem

ilről közvetlenül

Answ

Re

	Test	Got	Mark Fraction	
✓	`Ido t0;  // Van paraméter nélkül hívható konstruktora?`		0.5/0.5	✓
✓	`//Ido t0;` `//t0.show();  // Működik ashow? Jó értéket ír ki? (12:22)`	12:22	0.5/0.5	✓

✓	`//Ido t10(10, 20);  // Van kétparaméteres konstruktora?`
	`//t10.show();  // jó értéket ír ki? (10:20)`
		10:20	0.5/0.5	✓

✓	`// Adattagok privátak?`	OK	0.5/0.5	✓

✗	`//Ido t8(10, 30);`
	`//Ido t0 = t8 + 2;`
	`//t0.show();  // Jó az eredmény? 12:30`
	`//std::cout << std::endl;`
	`//t8.show();  // Bal oldal megváltozott? 10:30`
			0/1.5	✗

✗	`// dob kivételt?`		0/ 1	✗

✗	`//Ido t0(0, 1);  // tényleg oo:pp a kiírás formátuma?`
	`//t0.show();  // 00:01`
		0:1	0/ 1	✗

✗	`// Kontans objektumra is működik?`
	`//const Ido tc;`
	`//(tc + 1).show();  // 13:22`
			0/1.5	✗

```
1
2
3
4
5       int ora, perc;
6   public:
7       Ido(int ora = 12, int perc = 22) : ora(ora), perc(perc) {}
8       void show() const { std::cout << ora << ":" << perc; }
9   };
```

Házi feladat

INFOC INFÓ ▸ TANANYAG ▸ SEGÉDLET ▸ EXTRÁK ▸ BELÉPÉS

1. Játékok

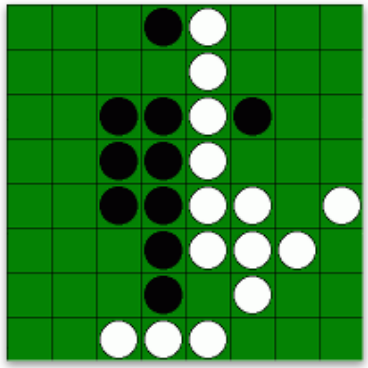
Amőba *

Készíts menüvezérelt C programot, mely amőbát játszik. A program legyen képes:

- a játék adminisztrálására egy tetszőleges méretű táblán,
- több támadó és védekező stratégiát alkalmazni,
- állás elmentésére és visszatöltésére.

Reversi **

Írj reversi játékot, amelyben a gép ellen lehet játszani! A program legyen képes:



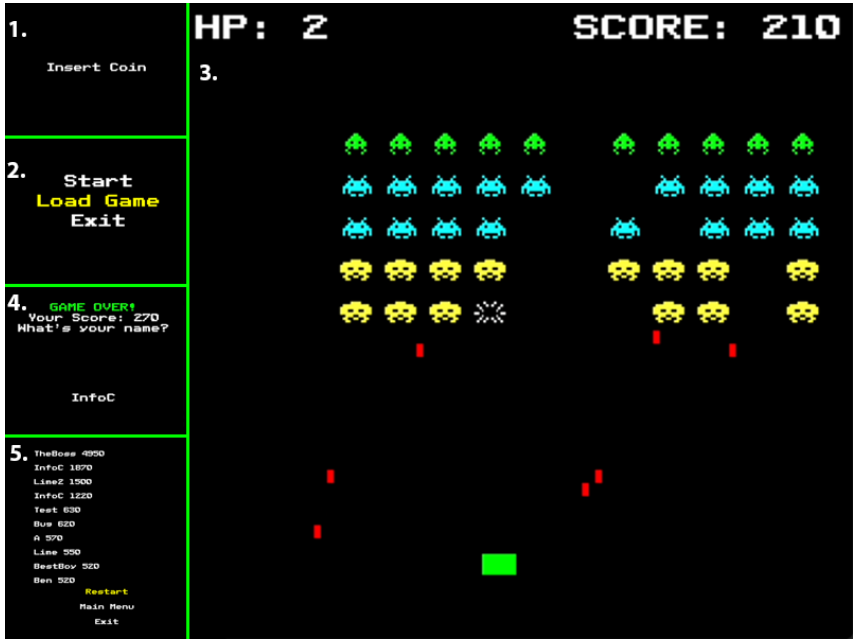
- Kirajzolni a pályát (szövegesen vagy grafikusan).
- Fájlba menteni és visszatölteni az állást.
- Ellenőrizni, hogy helyes-e a játékos lépése.
- Támadó stratégiákat alkalmazni.

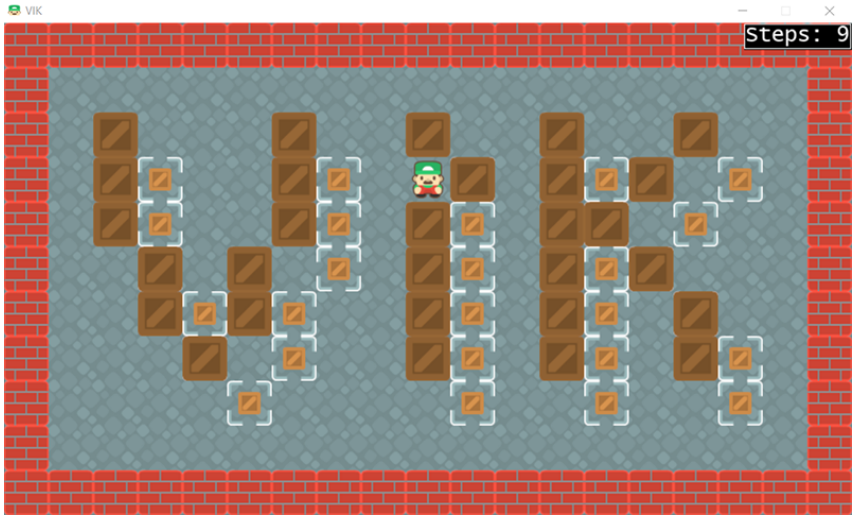
(Vigyázat, ennél is kötelező dinamikus memóriakezelést használni, bár az amőbával ellentétben ezt fix méretű pályán szokták játszani. Konzultálj a laborvezetővel!)

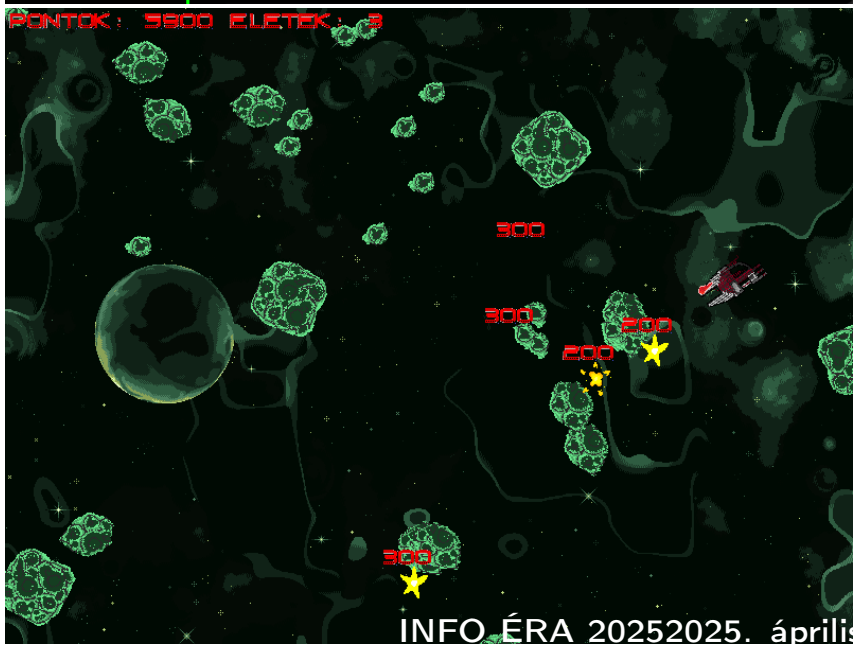
Hexxagon **

Táblás játék. A pálya hatszögletű elemekből áll. Mindkét fél néhány bábuval indul. Minden lépésben a

Házi feladat szépségverseny

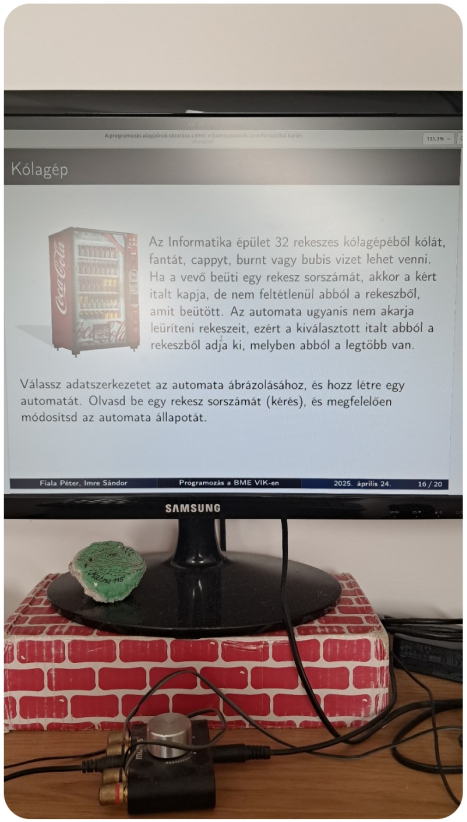






A jelen: ChatGPT a prog ZH-n

ChatGPT



Írj C programot erre a feladatra

ChatGPT

Ez a feladat egy 32 rekeszes italautomata modellezéséről szól. A lényeg:

- Minden rekesz tartalmaz egy adott fajta italt (pl. kóla, fanta, cappy, stb.)
- Ha valaki beüti egy rekesz sorszámát, akkor az ott tárolt ital fajtáját szeretné kérni.
- De: **nem biztos, hogy pont abból a rekeszből kapja**, amit beírt. Az automata **megkeresi azt a rekeszt, ahol ebből az italból a legtöbb van**, és **onnan adja ki**, hogy ne ürüljenek ki túl hamar a rekeszek.
- A program olvassa be a kérést (rekesz sorszáma), és **módosítsa az állapotot** (azaz csökkentse a megfelelő rekeszben a darabszámot).

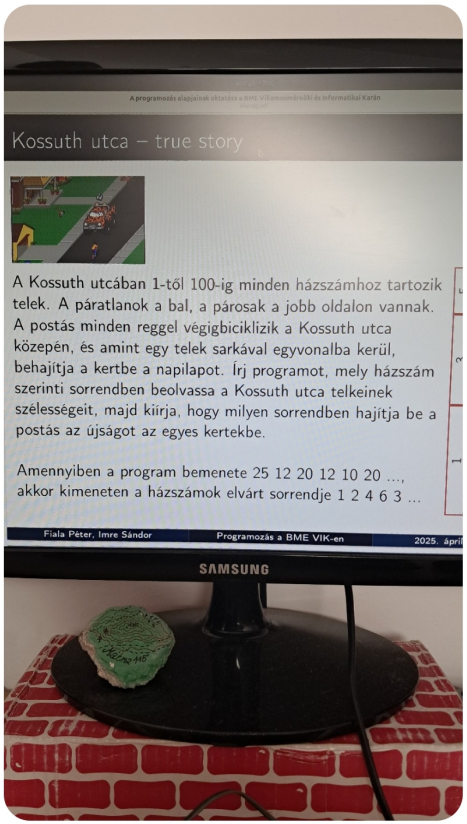
ChatGPT

C — Kód másolása

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

#define REKESZEK_SZAMA 32
#define ITALOK 6

const char* italok[] = {"kola", "fanta",
"cappy", "burnt", "bubis", "viz"};

typedef struct {
    char ital[20];
    int db;
} Rekesz;

int main() {
    Rekesz automata[REKESZEK_SZAMA];

    // Például minden rekeszbe
véletlenszerű ital + mennyiség
    for (int i = 0; i < REKESZEK_SZAMA;
i++) {
        strcpy(automata[i].ital,
italok[i % ITALOK]);
        automata[i].db = 5 + (i %
4);  // valami kezdő darabszám
    }

    int keresett_rekesz;
    printf("Adj meg egy rekesz szamot
(0-%d): ", REKESZEK_SZAMA - 1);
    scanf("%d", &keresett_rekesz);
```

A jelen: ChatGPT a Kossuth utcában

ChatGPT

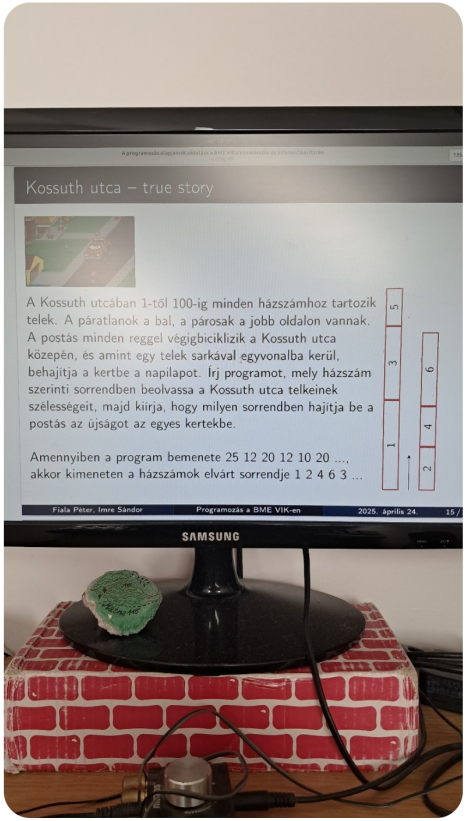


Írj C programot erre a feladatra

A feladat alapján egy olyan C programot kell írni, amely a Kossuth utca házainak

A jelen: ChatGPT a Kossuth utcában

ChatGPT



Köszönöm. Tökéletes. Újraküldöm postásos feladatot egy magyarázó ábrával. Azon látod, hogy a szélesség hogyan befolyásolja a sorrendet.

ChatGPT

Nagyon szuper, köszi az ábrát, így már teljesen egyértelmű!

A feladat lényege tehát:

- A postás **egyenesen halad az utca közepén**.
- Amikor egy telek **szélessége leér**, akkor **mindkét oldalra** (balra és jobbra) is **bedobja az újságot**, ha van ott ház.
- A páros számú házak a **jobb**, a páratlanok a **bal** oldalon vannak.
- Az input egy sor **szélesség**, felváltva bal és jobb telek (1-es ház bal, 2-es ház jobb, 3-as ház bal, 4-es ház jobb, stb.).
- A sorrendet az dönti el, **mikor ér véget** egy-egy telek, azaz mikor van a postás a telek közepénél.

Megoldás C nyelven:

ChatGPT

C — Kód másolása

```c
#include <stdio.h>

#define MAX 100

typedef struct {
    int hazszam;
    int tav;
} Haz;

int main() {
    int szelesseg;
    int bal_index = 0, jobb_index = 0;
    int bal_tav = 0, jobb_tav = 0;
    Haz hazak[MAX];
    int haz_db = 0;
    int hazszam = 1;

    // Szélességek olvasása
    while (scanf("%d", &szelesseg) == 1)
{
        if (hazszam % 2 == 1) { // bal
oldal (páratlan házszám)
            bal_tav += szelesseg;
            hazak[haz_db].hazszam =
hazszam;
            hazak[haz_db].tav = bal_tav;
        } else { // jobb oldal (páros
házszám)
            jobb_tav += szelesseg;
            hazak[haz_db].hazszam =
hazszam;
            hazak[haz_db].tav =
jobb_tav;
        }
        haz_db++;
        hazszam++;
```

MI – szabályozás

- Nem tiltó, hanem megengedő hozzáállás
- Minden tantárgy maga jelöli ki, hogy mire javasolja, mire nem
- A hallgató a beadandó feladatokról nyilatkozik
- A felelősség a hallgatóé
- Hangsúly a beadandó feladatok megvédésén
- Kreditet szerezni csak ellenőrzött körülmények között végzett munkával lehet

Összefoglalás

- Programozni minden mérnöknek tudnia kell
- A progalap tantárgyak programozói szemléletet és erős alapokat adnak
- Sikeres teljesítésük nagyon sok befektetett munkát igényel
- A hallgató minden támogatást megkap az önálló munkához
- Van lemorzsolódás, de a hallgatói visszajelzések pozitívak

Összefoglalás

- Programozni minden mérnöknek tudnia kell
- A progalap tantárgyak programozói szemléletet és erős alapokat adnak
- Sikeres teljesítésük nagyon sok befektetett munkát igényel
- A hallgató minden támogatást megkap az önálló munkához
- Van lemorzsolódás, de a hallgatói visszajelzések pozitívak

- Köszönöm szépen a figyelmet