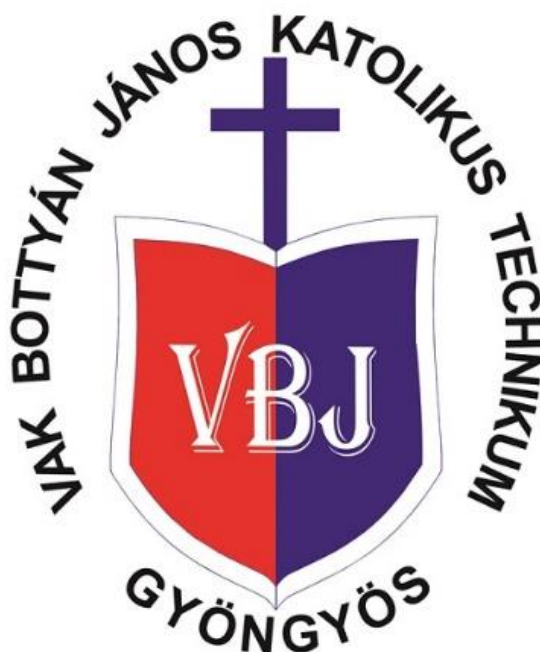


Dusza Árpád
Országos Programozói Emlékverseny

2024/25

VBJCode csapat

Vak Bottyán János
Katolikus Műszaki és Közgazdasági Technikum,
Gimnázium és Kollégium



Tartalom

Bevezetés.....	3
Felhasználói dokumentáció.....	3
1. A program indítása	3
2. Egy lehetséges futtatási folyamat	4
3. Grafikus felület használata	6
4. Kilépés a programból.....	6
Fejlesztői dokumentáció.....	6
1. Feladat leírása	6
2. Adatszerkezet leírása.....	6
3. A feladatrészek megoldása, leírása	7

Bevezetés

2023-ban iratkoztunk be a Vak Bottyán Katolikus Műszaki Technikumba Szoftverfejlesztő és tesztelő szakirányba. Mindannyian a 10.B osztályba járunk. Az informatika érdeklődési területe közös mindannyiunkban. Szívesen foglalkozunk az iskolában és az iskola idején kívül is vele. A versenyre azért jelentkeztünk, hogy kipróbáljuk magunkat és tanuljunk a versenyhelyzet kihívásaiból. A decemberi alapvizsga alkalmából eddig számunkra szokatlan nehézségű feladattal találkoztunk, ami kihívást jelentett számunkra. Ahogy egyre inkább törekedtünk a tökéletesebb megoldás felé egyre többet tanultunk a munkamegosztásról és a programozásról is. Reméljük munkánk a versenybizottság szerint is megfelel a kiírt feladatnak, amit a kövvetkezőkben részletezünk.

Felhasználói dokumentáció

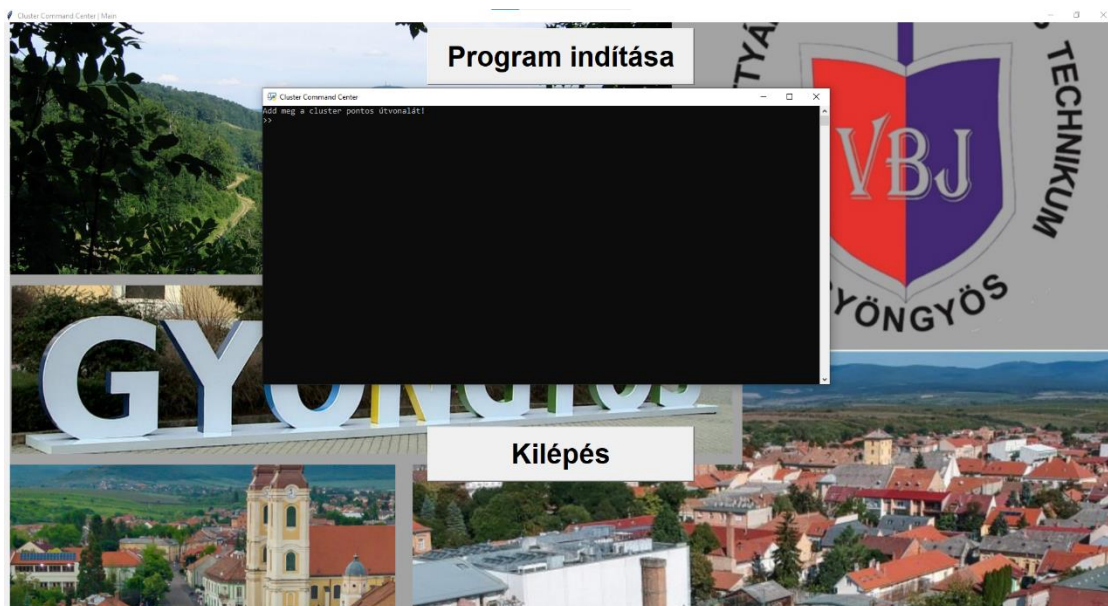
1. A program indítása

- Program indítása: Elindítja a programot, majd a cluster útvonalát kéri.
- Csapattagok: Versenyzők adatai olvashatók itt.
- README: Rövid leírást ad a program további használatáról a readme.txt fájlban.
- Dokumentáció: A program dokumentációja, a VBJCode Dokumentáció.pdf
- Kilépés: Kilép a programból.



2. Egy lehetséges futtatási folyamat

A képen a program indítása gombra nyomtunk. Elindult a program. Beolvasásnak a cluster abszolút elérési útvonalát kéri.

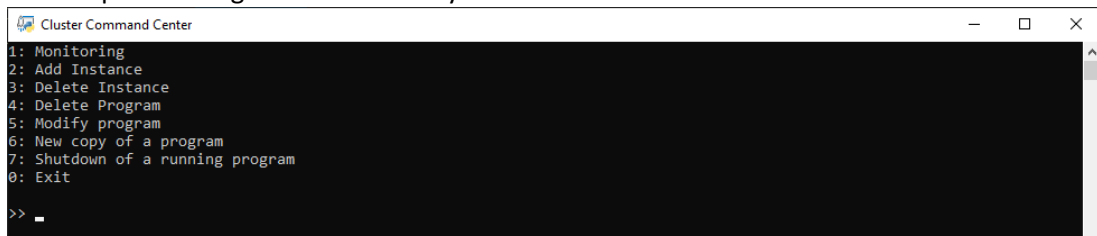


A cluster abszolút elérési útvonalát kell bemásolni.



A program további használata egy konzolos menü segíti.

A menüpontok megfelelnek a versenyfeladat kérdéseinek.



```
Cluster Command Center
szamitogep1
MAX      Elérhető
3500     3420
10000    9850

szamitogep2
MAX      Elérhető
3500     3250
10000    9500
```

```
Cluster Command Center
pelda
MAX      Elérhető
12000    12000
15000    15000

szamitogep1
MAX      Elérhető
3500     3420
10000    9850

szamitogep2
MAX      Elérhető
3500     3250
10000    9500
```

```
Cluster Command Center
1: pelda
2: szamitogep1
3: szamitogep2

Add meg a számítógép ID-jét:
>> 1
```

```
Cluster Command Center
Mit töröl?
>> chrome
chrome törölve
>> _
```

```
Cluster Command Center
Milyen programot akarsz módosítani?
>> word_
```

```
Cluster Command Center
Mennyi legyen a minimum futtatandó példányok száma?
>> 1_
```

```
Cluster Command Center
Melyik számítógépen akarsz futtatni az új peogrampéldányt?
1: pelda
2: szamitogep1
3: szamitogep2
>> 1_
```

```
Cluster Command Center
Add meg a számítógép nevét!
>> pelda
Add meg a millimagok számát és memóriakapacitást (MB) szóközzel elválasztva.
>> 12000 15000_
```

```
Cluster Command Center
Melyik programból akarsz egy új példányt futtatni?
1: chrome
2: word
>> 1_
```

```
Cluster Command Center
Melyik számítógépen akarsz leállítani a programpéldányt?
1: pelda
2: szamitogep1
3: szamitogep2
>> 1_
```

```
Cluster Command Center
Melyik számítógépen akarsz leállítani a programpéldányt?
1: pelda
2: szamitogep1
3: szamitogep2
>> 1
1: chrome-ixttru
2: chrome-rhnuab
3: word-vbxltv
>> 2
Program leállítva_
```

3. Grafikus felület használata

A grafikus felületen a program indításán és a csapattagok megjelenítésén túl a felülettel kapcsolatos összes információ gombok segítségével elérhető.

4. Kilépés a programból

A kilépés a programból a grafikus felületet zárja be, a feladatot megoldó konzolos program tovább használható.

Fejlesztői dokumentáció

1. Feladat leírása

A feladat leírása a versenyfeladat kiírása szerinti, a mellékelt linken található.

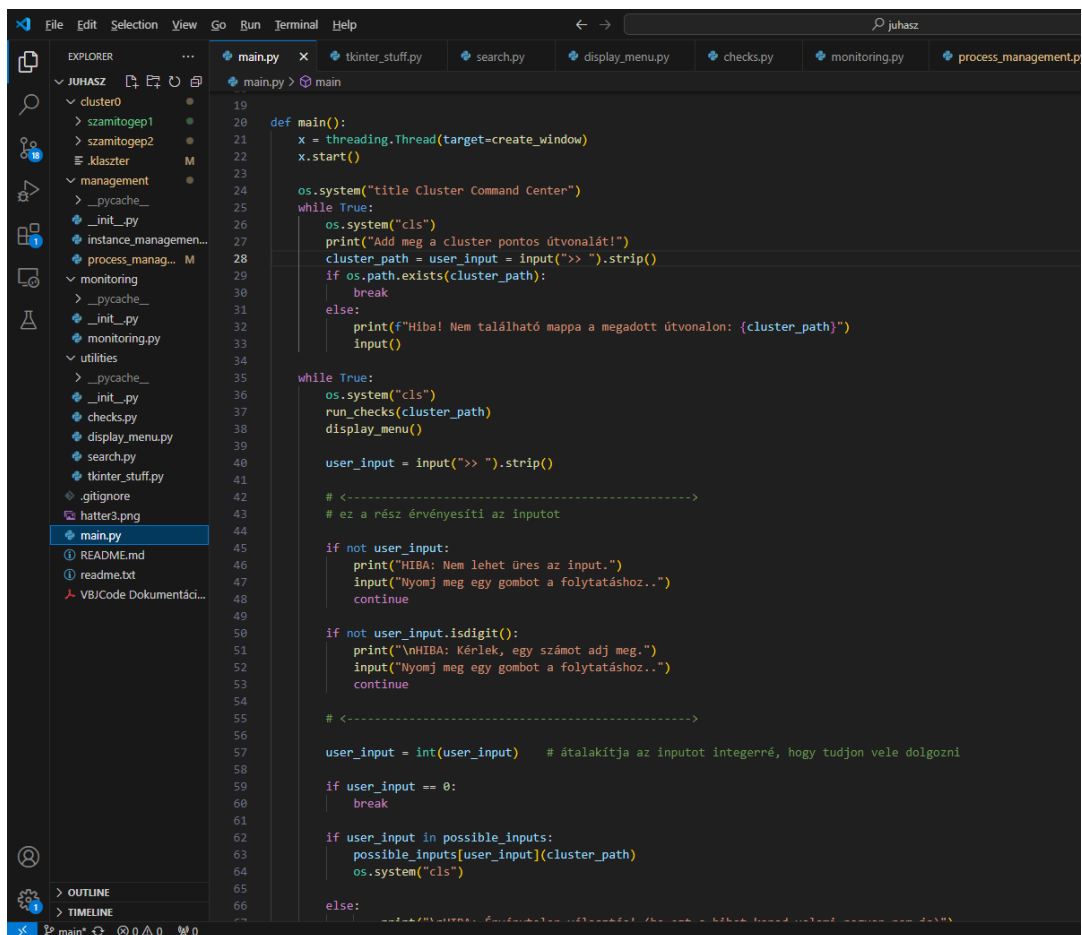
[Versenyfeladat leírás](#)

2. Adatszerkezet leírása

A feladat megvalósítását függvényekkel oldottuk meg, ezért a projektben 13 függvény van.

Név szerint: main, get_computer, get_cluster_data, get_computer_data, display_menu, run_checks, monitoring, add_instance, delete_instance, program_leallitas, program_modositas, uj_peldany, peldany_leallitas.

Ezek közül az alábbiakat emelnénk ki



```
def main():
    x = threading.Thread(target=create_window)
    x.start()

    os.system("title Cluster Command Center")
    while True:
        os.system("cls")
        print("Add meg a cluster pontos útvonalát!")
        cluster_path = user_input = input(">> ").strip()
        if os.path.exists(cluster_path):
            break
        else:
            print(f"Hiba! Nem található mappa a megadott útvonalon: {cluster_path}")
            input()

    while True:
        os.system("cls")
        run_checks(cluster_path)
        display_menu()

        user_input = input(">> ").strip()

        # <----->
        # ez a rész érvényesíti az inputot

        if not user_input:
            print("HIBA: Nem lehet üres az input.")
            input("Nyomj meg egy gombot a folytatáshoz..")
            continue

        if not user_input.isdigit():
            print("\nHIBA: Kérlek, egy számot adj meg.")
            input("Nyomj meg egy gombot a folytatáshoz..")
            continue

        # <----->

        user_input = int(user_input) # átalakítja az inputot integerré, hogy tudjon vele dolgozni

        if user_input == 0:
            break

        if user_input in possible_inputs:
            possible_inputs[user_input](cluster_path)
            os.system("cls")
        else:
```

3. A feladatrészek megoldása, leírása

Főprogram

A menü megjelenítése a függvények megjelenítését és futtatását valósítja meg. Elöltesztelő ciklusok segítségével az adatbekérést és kiíratást és az adatellenőrzést is megvalósítja.

```
def main():
    x = threading.Thread(target=create_window)
    x.start()

    os.system("title Cluster Command Center")
    while True:
        os.system("cls")
        print("Add meg a cluster pontos útvonalát!")
        cluster_path = user_input = input(">> ").strip()
        if os.path.exists(cluster_path):
            break
        else:
            print(f"Hiba! Nem található mappa a megadott útvonalon: {cluster_path}")
            input()

    while True:
        os.system("cls")
        run_checks(cluster_path)
        display_menu()

        user_input = input(">> ").strip()

        # <----->
        # ez a rész érvényesíti az inputot

        if not user_input:
            print("HIBA: Nem lehet üres az input.")
            input("Nyomj meg egy gombot a folytatáshoz..")
            continue

        if not user_input.isdigit():
            print("\nHIBA: Kérlek, egy számot adj meg.")
            input("Nyomj meg egy gombot a folytatáshoz..")
            continue

        # <----->
```

Run check függvény

A klaszterek ellenőrzése, hibák észlelése ebben történik

```
4 def run_checks(cluster_path):
5     cluster_config = os.path.join(cluster_path, ".klaszter")
6     running_processes = {}
7
8     if not os.path.isfile(cluster_config):
9         print("Hiba: Nincs cluster config fájl!")
10        return
11
12    required_processes = search.get_cluster_data(cluster_config)
13
14    computers = [computer for computer in os.listdir(cluster_path) if computer != ".klaszter"]
15
16    for computer in computers:
17        computer_path = os.path.join(cluster_path, computer)
18        cpu_usage = 0
19        ram_usage = 0
20        active = 0
21        processes = [process for process in os.listdir(computer_path) if process != ".szamitogep_config"]
22
23        config_path = os.path.join(cluster_path, computer, ".szamitogep_config")
24
25        if not os.path.isfile(config_path):
26            print(f"Hiba: A(z) {computer} számítógépben nem található config fájl!")
27            input(">> ")
28            return
29
30        with open(config_path, encoding="utf-8") as config:
31            max_cpu, max_ram = int(config.readline().strip()), int(config.readline().strip())
32
33            for process in processes:
34                with open(os.path.join(cluster_path, computer, process), encoding="utf-8") as p:
35                    data = p.readlines()
36                    if data[1].strip() == "Aktív":
37                        active += 1
38                        cpu_usage += int(data[2])
39                        ram_usage += int(data[3])
40
41        if cpu_usage > max_cpu and ram_usage > max_ram:
42            print(f"Hiba! A(z) {computer} számítógépben túllépésre került a CPU és a RAM kapacitása.")
43            input()
44            return
45        elif cpu_usage > max_cpu:
46            print(f"Hiba! A(z) {computer} számítógépben túllépésre került a CPU kapacitása.")
47            input()
48            return
49        elif ram_usage > max_ram:
50            print(f"Hiba! A(z) {computer} számítógépben túllépésre került a RAM kapacitása.")
51            input()
52            return
```

Interface

A program kezelésének grafikus felülete, amely felhasználóbarátabbá teszi a programot.

```
def start_program(root):
    root.destroy()

def open_readme():
    os.startfile("readme.txt")

def open_documentation():
    os.startfile("VBJCode Dokumentáció.pdf")

def credits(root):
    window = Toplevel(root)
    window.title("Cluster Command Center | Credits")
    window.geometry("500x150")
    Label(window, text="Bánka Levente - Dokumentáció, Trello\nBerecz Balázs - Programozás, dokumentáció\nJuhász Nándor - Csapatkapitány, programozás", font=("Times New Roman", 15)).pack()

def exit_program():
    quit()

def create_window():
    root = Tk()
    root.title("Cluster Command Center | Main")
    root.geometry("1920x1080")

    szia = tkFont.Font(family='Helvetica', size=36, weight='bold')

    image = PhotoImage(file="hatter3.png")
    background_label = Label(root, image=image)
    background_label.place(x=0, y=0, relwidth=1, relheight=1)

    start_button = Button(root, text="Program Indítása", activebackground="gray", command=lambda: start_program(root), height=1, width=15, font=szia)
    start_button.pack(pady=10)

    credits_button = Button(root, text="Csapattagok", activebackground="gray", command=lambda: credits(root), height=1, width=15, font=szia)
    credits_button.pack(pady=10)

    readme_button = Button(root, text="README", activebackground="gray", command=open_readme, height=1, width=15, font=szia)
    readme_button.pack(pady=10)

    documentation_button = Button(root, text="Dokumentáció", activebackground="gray", command=open_documentation, height=1, width=15, font=szia)
    documentation_button.pack(pady=10)

    exit_button = Button(root, text="Kilépés", activebackground="gray", command=exit_program, height=1, width=15, font=szia)
    exit_button.pack(pady=225)

    root.protocol("WM_DELETE_WINDOW", exit_program)
```

Process management

Ez kezeli a futó programokat a klaszterben lévő számítógépeken.

```
def program_leallitas(cluster_path):
    clear()
    directories = [dir for dir in os.listdir(cluster_path) if dir != ".klaszter"]
    print("Mit töröl?")
    mit_torol = input(">>> ")
    for i in range(len(directories)):
        szamitogep_hely = cluster_path + f"\\{directories[i]}"
        dirs = [dir for dir in os.listdir(szamitogep_hely) if dir != ".szamitogep_config"]
        for ii in range(len(dirs)):
            if (dirs[ii].split("-")[0] == mit_torol:
                os.remove(f"{szamitogep_hely}\\{dirs[ii]}")

    klaster = cluster_path + "\\klaszter"
    config = open(klaster, "r+", encoding="UTF-8")
    config_sorok = config.readlines()
    config.truncate(0)
    config = open(klaster, "w", encoding="UTF-8")
    for i in range(int(len(config_sorok)/4)):
        program = []
        for ii in range(4):
            program.append(config_sorok[i*4+ii])
        if program[0].strip() != mit_torol:
            for ii in range(4):
                config.write(program[ii])
    print(f"{mit_torol} törölve")
    input(">>> ")

def program_modositas(cluster_path):
    clear()
    print("Milyen programot akarsz módosítani?")
    mit_modosit = input(">>> ")
    clear()
    print("Add meg a módosított millimagok számát és memóriakapacitást (MB) szóközzel elválasztva.")
    mennyire_modosit = input(">>> ").split()
    clear()
    millimag = mennyire_modosit[0]
    memoria = mennyire_modosit[1]
    print("Mennyi legyen a minimum futtatandó példányok száma?")
    futatando_peldanyok = input(">>> ")
    clear()

    directories = [dir for dir in os.listdir(cluster_path) if dir != ".klaszter"]
    for i in range(len(directories)):
        szamitogep_hely = cluster_path + f"\\{directories[i]}"
        dirs = [dir for dir in os.listdir(szamitogep_hely) if dir != ".szamitogep_config"]
        for ii in range(len(dirs)):
```


Instance management

A számítógépeket kezeli, törli, klaszterbe helyez gépeket.

```
def add_instance(cluster_path):
    os.system("cls")
    print("Add meg a számítógép nevét!")
    name = input(">> ")

    if not name:
        return

    print("Add meg a millimagok számát és memóriakapacitást (MB) szóközzel elválasztva.")
    resources = [int(x) for x in input(">> ").split()]
    valid_characters = list(string.ascii_letters) + list(string.digits)

    instance_path = os.path.join(cluster_path, name)

    print(instance_path)

    if os.path.isdir(instance_path):
        print(f"Ilyen nevű számítógép már létezik! ({name})")
        input(">> ")
        return

    for c in name:
        if c not in valid_characters:
            print(f"Hiba! Helytelen karakter: {c}")
            input(">> ")
            return

    os.makedirs(f"{cluster_path}/{name}")
    with open(f"{cluster_path}/{name}/szamitogep_config", 'x') as file:
        file.write(f"{resources[0]}\n{resources[1]}")

def delete_instance(cluster_path):
    os.system("cls")
    computer = search.get_computer(cluster_path)
    computer_path = os.path.join(cluster_path, computer)

    processes = search.get_computer_data(computer_path, "processes")

    if processes:
        print(f"Hiba: Létezik futó folyamat a számítógépen!\n")
        print("\n<-----\n")
        for process in processes:
            with open(os.path.join(cluster_path, computer, process), encoding="utf-8") as p:
                data = p.readlines()
                print(f"Név:\t{process}\nIndítás ideje:\t{data[0].strip()}\nStátusz:\t{data[1].strip()}\n")
            print("<-----\n")
        else:
            print(f"\nGép törölve: {computer}")
```

Monitoring

A számítógépek állapotát, elérhető erőforrását írja ki

```
def monitoring(cluster_path):
    os.system("cls")
    directories = [dir for dir in os.listdir(cluster_path) if dir != ".klaszter"]

    for dir in directories:
        print(dir)
        config_path = os.path.join(cluster_path, dir, ".szamitogep_config")

        if not os.path.isfile(config_path):
            print(f"Hiba! Nincs konfigurációs fájl itt: {config_path}")
            input()
            exit()

        with open(config_path) as config_file:
            config = [int(line.strip()) for line in config_file]

        files = [file for file in os.listdir(os.path.join(cluster_path, dir)) if file != ".szamitogep_config"]

        print(f"\tMAX \tElérhető")

        for j in range(0, len(config), 2):
            max_cpu, max_ram = config[j], config[j + 1]
            cpu_usage = 0
            ram_usage = 0

            for file in files:
                file_path = os.path.join(cluster_path, dir, file)
                with open(file_path) as file:
                    data = file.readlines()
                    cpu_usage += int(data[2].strip())
                    ram_usage += int(data[3].strip())

            print(f"\t{max_cpu}\t{max_cpu - cpu_usage}")
            print(f"\t{max_ram}\t{max_ram - ram_usage}")

        print()
    input()
    return
```