A programozás alapjai 2.

Házi Feladat Dokumentáció

**Egy túrórudi gyár managelése**

Készítette: Papp Dominik Edvárd, EAT3D9

domi.papp55@gmail.hu

Készítés féléve: 2020/21/2

Tartalomjegyzék

[Felhasználói dokumentáció 3](#_Toc71824849)

[Osztályok statikus leírása 3](#_Toc71824850)

[RudiMeret (enum class) 3](#_Toc71824851)

[Felelőssége 3](#_Toc71824852)

[Attribútumok 3](#_Toc71824853)

[Metódusok 3](#_Toc71824854)

[Time 4](#_Toc71824855)

[Felelőssége 4](#_Toc71824856)

[Attribútumok 4](#_Toc71824857)

[Metódusok 4](#_Toc71824858)

[Rendeles 5](#_Toc71824859)

[Felelőssége 5](#_Toc71824860)

[Attribútumok 5](#_Toc71824861)

[Metódusok 5](#_Toc71824862)

[TeljesRendeles 7](#_Toc71824863)

[Felelőssége 7](#_Toc71824864)

[Attribútumok 7](#_Toc71824865)

[Metódusok 7](#_Toc71824866)

[RudiGyar 8](#_Toc71824867)

[Felelőssége 8](#_Toc71824868)

[Attribútumok 8](#_Toc71824869)

[Metódusok 9](#_Toc71824870)

[UML osztálydiagramm 12](#_Toc71824871)

[12](#_Toc71824872)

[Összegzés 13](#_Toc71824873)

[Mit sikerült és mit nem sikerült megvalósítani a specifikációból? 13](#_Toc71824874)

[Mit tanultál a megvalósítás során? 13](#_Toc71824875)

[Továbbfejlesztési lehetőségek 13](#_Toc71824876)

# Felhasználói dokumentáció

A program a gyártási folyamat teljeskörű managelésére alkalmas.

Használata és logikai felépítése a következő:

A Rendeles osztály objektumaival lehet külön, kis rendeléseket (rész rendeléseket) kezelni. A TeljesRendeles osztály objektumaival lehet olyan teljes rendeléseket kezelni, amelyek kis rendeléseket tartalmaznak. A RudiGyar osztály reprezentálja a gyárat, amelyen dolgozunk. A Time osztály objektumaival lehet dátumokat megadni.

Ha szeretnénk rendeléseket felvetetni a gyárral akkor ahhoz előbb létre kell hozni Rendeles objektumot/objektumokat és TeljesRendeles objektumot/objektumokat. Ha csak egy rendelést veszünk fel, akkor azt a rendelést fel kell vetetni a TeljesRendeles objektummal. Amennyiben több kis rendelés is van, abban az esetben a gyár makeRendelesArray függvényének segítségével egy vektorba kell olvasztani a rendeléseket és úgy átadni a TeljesRendeles objektumnak. A makeRendelesArray meghívása nem szükséges, az egy vektorba való olvasztás enélkül is abszolválható. Miután megvan a TeljesRendeles objektumunk (leszállítási dátum beállításával együtt) akkor az insertTeljesRendeles függvény segítségével felkérhetjük a gyárat, hogy vegye fel a rendelést. A gyár akkor fogja felvenni a rendelést, ha az utolsó rendelés legyártási után még a leszállítási idő előtt le tudja gyártani. Amennyiben ezt nem tudja megcsinálni, nem veszi fel. A gyárat megkérhetjük, hogy optimalizálja a rendelések sorrendjét. Ekkor a gyár igyekszik megtalálni azt a sorrendet, amellyel a legrövidebb idő alatt le tudja gyártani az összes rendelést. Ha egy optimalizálás után sem tudja felvenni a gyár a rendelést, akkor muszáj a rendelés leszállítási dátumán lazítani. A gyárat meg lehet kérni arra is, hogy minden sikeres felvétel után optimalizálja a rendelést.

A program használatának szintaktikájáról részletes bemutatást talál a program main.cpp fájljában.

# Osztályok statikus leírása

## RudiMeret (enum class)

### Felelőssége

A lehetséges túrórudi méretek kezelését segíti elő. Számok helyett a nevükkel lehet hivatkozni a méretekre.

### Attribútumok

#### Privát

* kicsi
* kozepes
* nagy

### Metódusok

#### Osztályon kívüli, de ehhez az osztályhoz tartozó

std::ostream& operator<<(std::ostream& os, const RudiMeret& right);

A megadott output streamre kiírja szövegesen az enum típusát.

## Time

### Felelőssége

A program időkezeléséért felelős. Év-hó-nap-óra megvalósítással működik. Minden hónap 30 nap. Csak egész órákat és az alapméretezett évnél nagyobb éveket értelmez kivéve létrehozáskor. Olyankor bármely nemnegatív számmal lehet létrehozni. Negatív időt nem értelmez.

### Attribútumok

#### Privát

* unsigned year éveket tartja számon, alapméretezetten: 2021, ez állítható
* unsigned month hónapokat tartja számon, alapméretezetten: 4, ez állítható
* unsigned day napokat tartja számon, alapméretezetten: 1, ez állítható
* unsigned hour órákat tartja számon, csak egészeket értelmez, alapméretezetten: 0, ez állítható

### Metódusok

#### Publikus

Time(unsigned year= currentYear, unsigned month=currentMonth, unsigned day = currentDay, unsigned hour = currentHour);

Alapméretezett és paraméteres konstruktor is. Alapméretezetten a privát attribútumok alapméretezett értékeit veszi fel.

bool setYear(unsigned set);

Beállítja az évet, ha a beállítandó érték legalább akkora, mint az alapméretezett év.

bool setMonth(unsigned set);

Beállítja a hónapot, ha a beállítandó érték legalább egy és felfeljebb 12.

bool setDay(unsigned set);

Beállítja a napot, ha a beállítandó érték legalább egy és felfeljebb 30.

bool setHour(unsigned set);

Beállítja az órát, ha a beállítandó érték nagyobb legalább egy és felfeljebb 23, a 24 órát 0 óraként értelmezi az osztály.

unsigned getYear()const;

Visszaadja az évet.

unsigned getMonth()const;

Visszaadja a hónapot.

unsigned getDay()const;

Visszaadja a napot.

unsigned getHour()const;

Visszaadja az órát.

unsigned getInHours()const;

Visszaadja az időt órában.

bool operator<(const Time& right)const;

Dátumot dátummal összehasonlító függvény.

bool operator>(const Time& right)const;

Dátumot dátummal összehasonlító függvény.

Time operator+=(const unsigned hour);

A bal oldali operandus dátumát órával adja össze.

Time operator+=(const Time& right);

A bal oldali operandus dátumát dátummal adja össze.

#### Osztályon kívüli, de ehhez az osztályhoz tartozó

std::ostream& operator<<(std::ostream& os, const Time& right);

A megadott output streamre kiírja szövegesen a dátumot.

## Rendeles

### Felelőssége

Ez az osztály kezeli az adott méretű túrórudira vonatkozó rendelést. A rendelésnek külön nevet lehet adni (pl. „Imre bacsi rendelese”, vagy „123”). Alapméretezetten a „Resz rendeles” nevet kapja. A rendeléshez olyan dátum társítható, amely tartalmazza, hogy legkésőbb meddig kell legyártódnia a rendelésnek.

### Attribútumok

#### Privát

* Time leszallitasiDatum legkésőbb eddig kell legyártódnia a rendelésnek
* RudiMeret rendeltRudi tartalmazza, hogy milyen méretű túrórudit kell gyártani
* unsigned mennyiseg ennyi darab túrórudit kell legyártani
* std::string ID a rendelés egyedi vagy alapméretezett nevét tárolja

### Metódusok

#### Publikus

Rendeles(RudiMeret rendeltRudi = RudiMeret::kicsi, unsigned db = 0, std::string ID = "Resz rendeles", Time leszallitasiDatum = Time());

Alapméretezett és paraméteres konstruktor is. Alapméretezetten a gyártani kívánt túrórudi mérete kicsi, mennyisége nulla, neve „Resz rendeles” és leszállítási dátuma az alapméretezett dátum.

Rendeles(const Rendeles& right);

Copy konstruktor.

RudiMeret getRendeltRudi()const;

Visszaadja a rendelt túrórudi méretét.

void setRendeltRudi(const RudiMeret& rendeltRudi);

Beállítja a rendelt túrórudi méretét.

int getMennyiseg()const;

Visszaadja a rendelt túrórudi mennyiségét.

void setMennyiseg(const unsigned mennyiseg);

Beállítja a rendelt túrórudi mennyiségét.

Time getLeszallitasiDatum()const;

Visszaadja, hogy meddig kell leszállítani a rendelést.

bool setLeszallitasiDatum(const Time& leszallitasiDatum);

Beállítja, hogy meddig kell leszállítani a rendelést. Csak akkor fogadja el a dátumot, ha nagyobb a jelenlegi dátumnál.

std::string getID()const;

Visszaadja a rendelés nevét, id-ját.

void setID(const std::string& ID);

Beállítja rendelés nevét, id-ját.

Time teljesitesHossza()const;

Visszaadja a rendelés teljesítéséhez szükséges időt átállítással.

bool operator<(const Rendeles& right)const;

Leszállítási idő szerinti összehasonlító függvény.

bool operator>(const Rendeles& right)const;

Leszállítási idő szerinti összehasonlító függvény.

#### Osztályon kívüli, de ehhez az osztályhoz tartozó

std::ostream& operator<<(std::ostream&os, const Rendeles& right);

A megadott output streamre kiírja a rendelés privát attribútumait.

## TeljesRendeles

### Felelőssége

Ez az osztály olyan rendeléseket kezel, amely több részrendelésből áll. A megrendelő lehet, hogy több méretű túrórudit akar gyártani, úgyhogy ezek egy logikai helyre tartoznak. Az osztály több mint 3 részrendelést is tud egyszerre kezelni, de a 3 túrórudi méret miatt ez felesleges.

### Attribútumok

#### Privát

* static int nextId ő biztosítja, hogy minden teljes rendelés egyedi azonosítóval legyen ellátva
* int ID a teljes rendelés egyedi azonosítója, egész szám
* std::vector<Rendeles> reszRendelesek Rendeles osztályokat tartalmazó vektor, ezekből a részrendelésekből áll elő a teljes rendelés
* Time leszallitasiDatum legkésőbb eddig kell leszállítani minden részrendelést

### Metódusok

#### Publikus

TeljesRendeles(Time& leszallitasiDatum, std::vector<Rendeles>& reszRendelesek);

Standard paraméteres konstruktor. Az ID privát attribútumot a nextId-val a sorban következő egész számra állítja.

TeljesRendeles();

Default konstruktor, az alapméretezett leszállítási dátum a (0,0,0,0), az ID-t a paraméteres konstruktorhoz hasonlóan állítja be. reszRendelesek attribútuma üres.

TeljesRendeles(const TeljesRendeles& right);

Standard copy konstruktor. Az ID-t nem másolja, új egyedi azonosítót kap.

Time getLeszallitasiDatum()const;

Visszaadja, hogy meddig kell leszállítani a teljes rendelést.

bool setLeszallitasiDatum(const Time& leszallitasiDatum);

Beállítja, hogy meddig kell leszállítani a teljes rendelést. Csak akkor fogadja el a dátumot, ha nagyobb a jelenlegi dátumnál.

std::vector<Rendeles> getReszRendelesek()const;

Visszaadja a teljes rendelés részrendeléseinek a vektorát.

void setReszRendelesek(const std::vector<Rendeles>& reszrendelesek);

Beállítja a teljes rendelés részrendeléseinek a vektorát.

int getID()const;

Visszaadja a teljes rendelés ID-ját.

Time teljesitesHossza()const;

Visszaadja, hogy mennyi idő teljesíteni/legyártani minden részrendelést. Számításba veszi az átállítási időt is amennyiben szükséges.

void operator=(const TeljesRendeles& right);

Standard egyenlő operátor. Az ID-t nem másolja, új egyedi azonosítót kap.

#### Osztályon kívüli, de ehhez az osztályhoz tartozó

std::ostream& operator<<(std::ostream& os, const TeljesRendeles& right);

A megadott output streamre kiírja a teljes rendelés privát attribútumait.

## RudiGyar

### Felelőssége

Ez az osztály nagyban hozzájárul a manageléshez. Ez az osztály a gyár karmestere, minden lényeges tevékenységet ő lát el. Kezelni és manipulálni tud teljes- és részrendeléseket. Alkalmas std::vector<Rendeles>-ek manipulálására. Rendelkezik optimalizáló metódussal is mely állítható, hogy ez automatikusan megtörténjen-e teljes rendelések felvételekor. Az, hogy a gépek milyen méretű túrórudi gyártására vannak beállítva az az összes objektumra érvényes, statikus. Rendelkezi statikus metódusokkal is.

### Attribútumok

#### Privát

* static const unsigned kicsiKapacitas ennyi darab kicsi méretű túrórudit tud legyártani a gyár egy nap alatt
* static const unsigned kozepesKapacitas ennyi darab közepes méretű túrórudit tud legyártani a gyár egy nap alatt
* static const unsigned nagyKapacitas ennyi darab nagy méretű túrórudit tud legyártani a gyár egy nap alatt
* static const unsigned atallitasiIdo ennyi órába telik átállítani a gyár gépeit egy másik méretű túrórudi gyártására
* static RudiMeret currentState ilyen méretű túrórudi gyártására vannak jelenleg beállítva a gyár gépei. A többi osztálynak szüksége van erre az információra így muszáj, hogy statikus legyen.
* Time utolsoGyartasVege ekkora lesz legyártva az összes megadott rendelés
* std::vector<Rendeles> rendelesek a rendeléseket tároló vektor, Rendeles osztályokat tárol
* bool autoOptimalize ha értéke igaz akkor a gyár automatikusan optimalizálni fog, ha hamis, nem fog

### 

### Metódusok

#### Publikus

RudiGyar(Time utolsoGyartasVege = Time(), bool autoOptimalize = false);

Default és paraméteres konstruktor. Az utolsó gyártás végének időpontja az aktuális dátum (lásd Time osztály) és az automatikus optimalizálás alapméretezetten nincs beállítva.

RudiGyar(const RudiGyar& theOther);

Default copy konstruktor.

void insertTeljesRendeles(const TeljesRendeles& teljesRendeles);

Amennyiben le tudja gyártani a rendelést a határidőig akkor felveszi a teljes rendelést a rendelések közé. Amennyiben ezt nem tudja megvalósítani akkor azt kiírja a standard outputra. Sikeres felvétel esetén a beállított automatikus optimalizálás értékétől függően optimalizálja is a gyár rendeléseit.

void optimalize();

Ez az attribútum valósítja meg a gyártási folyamat optimalizálását. Általa optimalizál az insertTeljesRendeles attribútum is. Működése a következő: A rendelesek vektort szétszedi 3 különböző vektorra oly módon, hogy egy vektorban csak egy bizonyos méretre vonatkozó rendelések legyenek. Minden mérethez különböző vektor készül. Ezeket a vektorokat utána úgy rendezi, hogy leszállítási dátum szerinti növekvő sorrendben legyenek a rendelések. A rendezés után a 3 vektort egybeolvasztja a merge attribútum segítségével úgy, hogy a sorrend az új vektorban a kicsi, közepes, nagy legyen. Ennek oka, hogy a kicsi az alapméretezett gyártási méret, így megspórolva egy átállítást. Az optimalizálás helyessége érdekében a merge attribútumot mindig az alapméretezett mérethez kapcsolódó vektorral az első paraméterként kell meghívni. Az összeolvasztás után a végig nézi az elkészült vektort, hogy így mindent le lehet-e gyártani. Amennyiben nem, a problémás rendelést eggyel előrébb teszi a sorrendben és újra le ellenőrzi, hogy le lehet-e gyártani mindent időben. Ezt addig csinálja amíg az előbb említett kérdésre a válasz igen nem lesz. Az attribútum lefutása minden esetben garantált, hisz mindig lesz egy olyan sorrend, amely kielégíti a feltételeket mert az eredeti rendelés vektor alapból teljesíthető. Erről az insertTeljesRendeles attribútum gondolkodik. Hibakezelésre nincs szükség.

std::vector<Rendeles> makeRendelesArray(const Rendeles& r1, const Rendeles& r2, const Rendeles& r3) const;

3 Rendeles objektumból egy egybefüggő vector<Rendeles>-t csinál melyben az adatok r1 r2 r3 sorrendben követik egymást.

std::vector<Rendeles> makeRendelesArray(const Rendeles& r1, const Rendeles& r2) const;

1 Rendeles objektumból egy egybefüggő vector<Rendeles>-t csinál melyben az adatok r1 r2 sorrendben követik egymást.

std::vector<Rendeles> makeRendelesArray(const Rendeles& r1) const;

1 Rendeles objektumból vector<Rendeles>-t csinál.

static unsigned getKicsiKapacitas();

Visszaadja a gyár kicsi túrórudi gyártására vonatkozó napi kapacitást.

static unsigned getKozepesKapacitas();

Visszaadja a gyár közepes túrórudi gyártására vonatkozó napi kapacitást.

static unsigned getNagyKapacitas();

Visszaadja a gyár nagy túrórudi gyártására vonatkozó napi kapacitást.

static unsigned getAtallitasiIdo();

Visszaadja a gépek átállításához szükséges időt órában.

static RudiMeret getCurrentState();

Visszaadja, hogy a gépek jelenleg milyen méretű túrórudi gyártására vannak beállítva.

Time getUtolsoGyartasVege()const;

Visszaadja, hogy mikorra lesz legyártva minden rendelés.

std::vector<Rendeles> getRendelesek()const;

Visszaadja a rendelések vektorát.

unsigned getRendelesekSzama()const;

Visszaadja, hogy hány darab rendelést kell legyártani.

bool getAutoOptimalize()const;

Visszaadja, hogy egy teljes rendelés felvétele után automatikusan optimalizálódik-e a gyártás.

void setAutoOptimalize(bool autoOptimalize);

Beállítja, hogy egy teljes rendelés felvétele után automatikusan optimalizálódjon-e a gyártás.

void setCurrentState(const RudiMeret& currentState);

Beállítja a gyár gépeit egy bizonyos rudiméret gyártására.

std::vector<Rendeles> selectKicsi(const std::vector<Rendeles>& rendelesek) const;

A paraméterként átvett vektorból kiválasztja azokat a rendeléseket, amelyek kicsi méretű túrórudi gyártását kérik.

std::vector<Rendeles> selectKozepes(const std::vector<Rendeles>& rendelesek) const;

A paraméterként átvett vektorból kiválasztja azokat a rendeléseket, amelyek közepes méretű túrórudi gyártását kérik.

std::vector<Rendeles> selectNagy(const std::vector<Rendeles>& rendelesek) const;

A paraméterként átvett vektorból kiválasztja azokat a rendeléseket, amelyek nagy méretű túrórudi gyártását kérik.

std::vector<Rendeles> merge(const std::vector<Rendeles>& bal, const std::vector<Rendeles>& kozep, const std::vector<Rendeles>& jobb) const;

A három paraméterül átvett vector<Rendeles> vektort összeolvasztja bal kozep jobb sorrendben egy úgy vector<Rendeles> vektorrá, majd visszatér vele.

#### Osztályon kívüli, de ehhez az osztályhoz tartozó

std::ostream& operator<<(std::ostream& os, const RudiGyar& right);

A megadott output streamre kiírja a gyár privát attribútumait.

# UML osztálydiagramm

# 

# Összegzés

## Mit sikerült és mit nem sikerült megvalósítani a specifikációból?

Mindent sikerült megvalósítanom, amit a specifikációban leírtam. Annyi változás történt, hogy implementáltam egy optimalizálást is. Ezt a specifikációban nem jeleztem, hogy lesz.

## Mit tanultál a megvalósítás során?

Nagyon kellemes gyakorlás volt. Megtapasztaltam milyen jól alkalmazható programozási nyelv C++ és mennyire rugalmas. Minden felmerülő problémát viszonylag könnyen meg tudtam oldani a nyelvnek hála. Az objektum orientáltság és az egységbezárás nagyon hasznomra vált. Átláthatóan tudtam programozni, nem keveredtem bele a saját kódomba. Mindent egyértelműen szét lehet bontani logikailag egybetartozó részekre és ez nagyon megkönnyítette a dolgomat.

## Továbbfejlesztési lehetőségek

A két legfőbb továbbfejlesztési lehetőség az adatbázis kezelésének és egy menünek az implementálása. Egy másik lehetőség még az optimalizáló algoritmus hatékonyságának a növelése.

Az alkalmazási terület bővítésére Templatekkel kellene megvalósítani a programot és egy interface-t készíteni hozzá. Így szinte bármilyen gyár managelésére alkalmassá válna a program.