

# ООП Изпити

## Информатика 20/21

Въпрос 1

Отговорен

От максимално  
12,00

Отбелязване  
на въпроса

В задачата можете да използвате наготово класа `std::vector` от `<vector>`.

Маймуната Зимбу решил, че е крайно време да започне сериозен бизнес. Навремето, когато пристигнал в България, докато го карали към зоопарка, шофьорът минал прекалено близо до няколко университета. Така докато стигнат, Зимбу вече бил собственик на няколко бакалавърски дипломи и на две магистърски. Когато пристигнали в зоопарка, оттам му казали, че е прекалено квалифициран и не го пуснали вътре. Така той се оказал в сложна ситуация -- трябвало сам да си изкарва прехраната.

Зимбу разгледал дипломите си и особено му харесала една, издадена от "Вселенския университет за висши науки и съвършенство във всички области" от Горно Нанадолнище. На нея пишело: "ИТ специалист!" (с удивителна). И понеже сега "компютрите са модерни", Зимбу решил да продава компютри. Веднага си регистрирал фирма (ЕТ "Стратиджик инфлуенсинг енд трендсетинг – Зимбу Маймуната") и започнал дейността си в един гараж. Но Зимбу не бил вчерашен. Още с пристигането си тук, той бил разбрал, че не иска да работи и му се струвало съвсем естествено да не прави нищо, а да получава по много. Затова той иска да автоматизира дейността си и някой да му напише програма (разбира се, безплатно – "Тия програмисти какво само искат!"). С нея хората сами ще могат да си направят конфигурация и да си я продадат сами на себе си. Вашата задача е да помогнете на Зимбу.

**А) (2 точки)** Един компютър се състои от един или повече компоненти. Всички компоненти са `immutable` обекти. След като веднъж бъдат създадени, свойствата им не могат да се променят. Всеки компонент да има:

- етикет (`label`), който е `std::string` обект.
- функция `double price() const`, която връща цената на компонента. Тя ще се имплементира от всеки конкретен компонент и ще пресмята цената по някакво правило.
- функция `void output(std::ostream& out) const`, която извежда информация за компонента – неговото име и цена.

Програмата ви трябва да поддържа два конкретни компонента – процесор (`cpu`) и памет (`memory`).

Процесорът има брой ядра (`cores`) и честота в MHz (`clock speed`). И двете са числа от тип `unsigned short`. Броят на ядрата може да е между 1 и 8. (Зимбу много държи да не се мине и да не продаде някое ядро в повече). За честотата няма ограничение ("Нека има, да се радват клиентите!" казва Зимбу). Цената на процесора се смята като броя на ядрата, умножен по 29.99 лева.

Паметта има капацитет (`capacity`) в гигабайти (`unsigned short`). Трябва да е число между 1 и 10'000 (Зимбу лъже в обявите, защото клиентите му са лековерни). Цената на паметта е 89.99 лева на гигабайт.

Ако се опита да се създаде компонент с некоректно подадени параметри, да се хвърли изключение.

Реализирайте подходяща йерархия, за да представите горе-описаната ситуация. Проектирайте йерархията по всички добри практики, които познавате, като имате предвид, че Зимбу в бъдеще ще добавя и други видове компоненти. Например помислете какъв трябва да е деструкторът, кои операции да са `pure virtual` и кои да изнесете в базовия клас и т.н.

**Б) (2 точки)** Напишете `factory` функция `component* create_component()`. Тя трябва да попита потребителя какъв тип компонент иска да създаде – процесор или памет. След това тя въвежда съответните му характеристики, създава динамично (със `new`) обект от съответния тип и го връща. Ако потребителят въведе невалиден тип или създаването на обект е неуспешно, да се върне `nullptr`.

## Изпит Задачи

**Б) (2 точки)** Напишете factory функция `component* create_component()`. Тя трябва да попита потребителя какъв тип компонент иска да създаде – процесор или памет. След това тя въвежда съответните му характеристики, създава динамично (със `new`) обект от съответния тип и го връща. Ако потребителят въведе невалиден тип или създаването на обект е неуспешно, да се върне `nullptr`.

**В) (4 точки)** Напишете клас `configuration` представящ конфигурация. Класът трябва да може да съхранява в себе си един или повече компоненти. Няма ограничение за техния брой – сега имаме само два вида, но в бъдеще Зимбу ще добави и още.

Класът да има следните член-функции:

- `double price() const` – връща цената на конфигурацията. Пресмята се като сума от цените на всички компоненти.
- `std::size_t size() const` – връща броя на компонентите в конфигурацията.

За класа предефинирайте следните оператори:

- `operator[]`, който позволява да се достъпи един от компонентите на конфигурацията. Операторът да приема стойност от тип `std::size_t` – индекс на компонент (число между 0 и `configuration::size()-1`). Направете проверка дали подаденият индекс е коректен и ако това не е така, хвърлете изключение. Операторът трябва да е константна функция и да връща `const` reference. През него не трябва да може да се променят компонентите.
- `operator<<`, който извежда информация за конфигурацията в `std::ostream`. При извеждането най-напред да се изведат един по един всички компоненти, използвайки техните функции `output`. Накрая, да се изведе и още един ред с общата цена на конфигурацията.

Изграждането на конфигурацията ще бъде сложно и затова не трябва да може всеки да го прави. За целта:

- Дефинирайте default конструктора като `private`. Той да създава празна конфигурация, в която няма компоненти.
- Дефинирайте `private` функция `void insert(const component* c)`, която добавя нов компонент към конфигурацията. Функцията директно да съхранява указателя, без да прави копие на подадения компонент.
- Деструкторът на класа трябва да изтрива всички съхранени в него обекти.
- Предотвратете копирането на конфигурации, като направите копиращия конструктор и копиращото присвояване или да са `private` и без дефиниция, или да са изтрити функции.

Относно `insert` имаме и още едно изискване. В една конфигурация не може да има повече от един компонент от даден тип. Когато добавяте нов компонент, използвайте `RTTI/typeid`, за да проверите дали новият компонент няма същия тип като някой, който вече е добавен. Ако това е така, компонентът да не се добавя, а да се хвърли изключение.

**Г) (2 точки)** Напишете функция `create_configuration`. Тя трябва да създава и да връща `configuration` обект. Направете я `friend` за класа `configuration`.

Функцията да създава динамично нов `configuration` обект (със `new`), да въвежда за него един или повече компоненти от потребителя (чрез `create_component`), да ги добавя в конфигурацията (чрез `configuration::insert`) и накрая да връща готовия обект.

Ако по някаква причина създаването не успее, върнете `nullptr`.

**Д) (2 точки)** Напишете програма, която използва горните класове, за да направи следното:

1. Позволява на потребителя да въведе каквито пожелае компоненти
2. Изгражда по тях конфигурация
3. Извежда на екрана избраните от потребителя компоненти и цената на конфигурацията.
4. Същата информация да се изведе и в текстов файл с име посочено от потребителя.

# Изпит Теория

Защо работи инициализиращият лист?

Започнат на	вторник, 15 юни 2021, 09:00
Състояние	Завършен
Приключен на	вторник, 15 юни 2021, 09:34
Изминало време	34 мин. 17 сек.

Въпрос **1**  
Отговорен  
От  
максимално  
1,00  
✓  
Отбелязване  
на въпроса

Нека са дадени следните дефиниции:

```
class test {  
public:  
    test(int)  
    {}  
};  
  
void g(test)  
{}  
  
void f(int)  
{}
```

Маркирайте тези от изразите, които смятате, че ще се компилират.

- ☒ g(5.5);
- ☒ g(5);
- ☐ f(test(5));
- ☐ h(5);

Въпрос **2**  
Отговорен

За всяка от член-променливите посочете нейната видимост.

Въпрос **2**  
Отговорен  
От  
максимално  
1,00  
✓  
Отбелязване  
на въпроса

За всяка от член-променливите посочете нейната видимост.

```
class A {  
    int a1;  
private:  
    int a2;  
    int a3;  
public:  
    int a4;  
private:  
};  
  
struct B {  
    int b1;  
private:  
    int b2;  
    int b3;  
public:  
    int b4;  
private:  
};
```

- A::a1 private ⇅
- A::a2 private ⇅
- A::a3 private ⇅
- A::a4 public ⇅
- B::b1 public ⇅
- B::b2 private ⇅
- B::b3 private ⇅
- B::b4 public ⇅

Въпрос **3**  
Отговорен  
От  
максимално  
1,00  
🚩  
Отбелязване  
на въпроса

Какъв вид копиране извършва автоматично генериранят оператор за присвояване в дадения по-долу фрагмент?

```
class Test {  
    int var = 0;  
};  
  
int main()  
{  
    Test a, b;  
    a = b;  
    return 0;  
}
```

- ☐ Задължително е да се използва покомпонентно копиране с оператора за присвояване на всеки от членовете.
- ☐ Компиляторът не може да генерира оператор за присвояване.
- ☒ Компиляторът може да използва тривиално копиране (например с `memcpy` или `memmove`)

Въпрос **4**  
Отговорен  
От  
максимално  
1,00  
🚩  
Отбелязване  
на въпроса

Каква версия на оператора за присвояване ще генерира компиляторът за дадения по-долу фрагмент?

```
class Test {  
    int var = 0;  
    const int cvar = 0;  
};  
  
int main()  
{  
    Test a, b;  
    a = b;  
}
```

- ☐ `Test& operator=(Test)`
- ☐ `Test& operator=(Test &)`
- ☒ `Test& operator=(const Test &)`
- ☐ Компиляторът не може да генерира оператор за присвояване, защото класът съдържа константа.

<div>Въпрос <b>5</b></div> <div>Отговорен</div> <div>От максимално 1,00</div> <div>🚩 Отбелязване на въпроса</div>	<div>Ще даде ли грешка компилаторът, ако дефинираме оператора за присвояване на <b>Test</b> така:</div> <div><pre>class Test {     int var; public:     bool operator=(Test&amp; other)     {         if (this == &amp;other)             return false;         var = other.var;         return true;     } };</pre></div> <div>Изберете едно:</div> <div><input type="radio"/> Истина</div> <div><input checked="" type="radio"/> Лъжа</div>
<div>Въпрос <b>6</b></div> <div>Отговорен</div> <div>От максимално 1,00</div> <div>🚩 Отбелязване на въпроса</div>	<div>Възможно ли е в един клас да се дефинират няколко различни версии на оператора за присвояване?</div> <div>Изберете едно:</div> <div><input checked="" type="radio"/> Истина</div> <div><input type="radio"/> Лъжа</div>
<div>Въпрос <b>7</b></div> <div>Отговорен</div> <div>От максимално 1,00</div> <div>🚩 Отбелязване на въпроса</div>	<div>Кои от следните оператори можем да предефинираме?</div> <div>Всеки верен отговор увеличава точките за въпроса, а всеки погрешен ги намалява. В скоби до операторите са посочени техните имена.</div> <div><input type="checkbox"/> <b>?:</b> (ternary conditional)</div> <div><input checked="" type="checkbox"/> <b>-&gt;</b> (member of pointer)</div> <div><input checked="" type="checkbox"/> <b>new</b></div> <div><input type="checkbox"/> <b>.</b> (member access)</div>
<div>Въпрос <b>8</b></div> <div>Отговорен</div> <div>От максимално 1,00</div> <div>🚩 Отбелязване на въпроса</div>	<div>Посочете вярно ли е следното твърдение.</div> <div>"В C++, когато предефинираме оператор, можем да променим неговия приоритет."</div> <div>Изберете едно:</div> <div><input type="radio"/> Истина</div> <div><input checked="" type="radio"/> Лъжа</div>
<div>Въпрос <b>9</b></div> <div>Отговорен</div> <div>От максимално 4,00</div> <div>🚩 Отбелязване на въпроса</div>	<div>Обяснете следните неща свързани с виртуалните функции в C++:</div> <div><ol style="list-style-type: none"><li>Обяснете какво представляват виртуалните функции.</li><li>Защо са ни нужни те (какъв проблем решават)?</li><li>Обяснете какво се случва ако извикаме такава функция през обект, указател към обект или reference към обект.</li><li>Какво представляват чисто виртуалните (pure virtual) функции. Как наричаме класовете, в които има поне една такава функция. Какви особености им класове.</li></ol></div> <div>Дайте примери.</div>
<div>Въпрос <b>10</b></div> <div>Отговорен</div> <div>От максимално 4,00</div> <div>🚩 Отбелязване на въпроса</div>	<div>Обяснете шаблона синглетон в C++:</div> <div><ol style="list-style-type: none"><li>За какво се използва той (какво ни дава)?</li><li>Обяснете как работи.</li><li>Дайте пример.</li></ol></div>
<div>Въпрос <b>11</b></div> <div>Отговорен</div> <div>От максимално 4,00</div> <div>🚩 Отбелязване на въпроса</div>	<div>Обяснете следните неща свързани със статичните членове на клас C++:</div> <div><ol style="list-style-type: none"><li>По какво статичните член променливи се различават от обикновените членове на клас?</li><li>Какво са статичните член-функции? Какви техни разлики спрямо обикновените функции знаете? Как ги извикваме?</li><li>Кога се инициализират статичните член-променливи. Какъв е синтаксисът на C++ за това.</li></ol></div> <div>Дайте примери.</div>

# СИ 20/21 - I. поток

Позволено е използването на STL.

Спазвайте принципите на ООП. Придържайте се към добрите практики за писане на код.

*В задачата по-долу, всички описани мерки са дробни числа.*

Напишете клас **Rectangle**, описващ правоъгълник, със следните характеристики:

- дължина
- широчина
- цвят (низ)
- методи, връщащи стойностите на член-данните на класа
- метод за изчисляване на лице

Напишете клас **Circle**, описващ кръг, със следните характеристики:

- радиус
- цвят (низ)
- методи, връщащи стойностите на член-данните на класа
- метод за изчисляване на лице

Напишете клас **Shapes**, със следните характеристики:

- съдържа в себе си горе-дефинираните геометрични фигури
- метод за добавяне на нов кръг към колекцията
- метод за добавяне на нов правоъгълник към колекцията
- метод, който връща сумата от всички лица на правоъгълниците
- метод, който връща сумата от всички лица на кръговете
- метод, който връща геометрична фигура по подаден индекс

Не е позволено използването на STL.

Спазвайте принципите на ООП. Придържайте се към добрите практики за писане на код.

## NamedObject

Даден е клас **NamedObject**, който ще пази даден обект под някакво име. Всеки **NamedObject** ще съдържа и **id** (цяло, неотрицателно число), име (низ с произволна дължина) и обект (който може да бъде от произволен тип).

Създайте подходящ конструктор с параметри, както и подходящи методи за достъп до член-данните на класа. Стойностите на **NamedObject** се задават само при създаването на обекта.

## NamedObjectArray

Даден е клас **NamedObjectArray**, който ще пази неограничен (до колкото стига паметта) брой инстанции от **NamedObjectArray**.

Създайте метод за добавяне на нов **NamedObject** обект към колекцията. Създайте метод, който връща броя обекти в колекцията. Предефинирайте оператора **[]**, който да връща обекта, съответстващ на даден индекс. Ако индекса е невалиден, да се хвърли изключение **out\_of\_range**

СИ 20/21 - II. Поток

13:51

LTE1 3%



## Преглед



## Въпрос 1

Частично правилен отговор  
7,00 от максимално 10,00 точки

Какво ще изведе на екрана следната програма ?  
Обяснете защо е изведен всеки ред от програмата.

```
#pragma once
#include <iostream>

class Foo {
public:
    Foo() {
        std::cout << "Foo()\n";
    }

    Foo(const Foo& other) {
        std::cout << "Foo copy()\n";
    }

    Foo& operator=(const Foo& other) {
        if(this != &other) {
            std::cout << "Foo=\n";
        }
        return *this;
    }
};

class Bar: public Foo {
public:
    Bar() {
        std::cout << "Bar()\n";
    }

    Bar(const Bar& other) {
        std::cout << "Bar copy()\n";
    }

    Bar& operator=(const Bar& other) {
        if(this != &other) {
            this->Foo::operator=(other);
            std::cout << "Bar=\n";
        }
        return *this;
    }
};

int main() {
    Bar a;
    Foo b = a;

    Foo c;
    c = b;

    Bar d;
    d = a;
    return 0;
}
```





13:51

LTE 3%



## Преглед



Създаването на с вика Foo(); Когато вече имаме създадени два обекта и им сложим = викаме оператор= на съответния клас Foo=; и Foo(); При създаването на Bar d се извиква Bar(); d=a вече създадени обекти извикват оператор = : Bar=;

## Въпрос 2

Частично правилен отговор  
6,00 от максимално 10,00 точки

В кода има 5 грешки. Какви са те ? Предложете начин за тяхното оправяне (с три-четири думи, или ред код).

```
#pragma once
#include <iostream>
#include <string>

class Car {
public:
    Car(): model(""), horsepower(0) {}

    void get_model() const {
        std::cout << this->model;
    }

    void* set_model(const char* model) const {
        delete[] this->model;
        this->model = new char(strlen(model) + 1);
        strcpy(this->model, model);
    }

    ~Car() {
        delete[] this->model;
        delete this->horsepower;
    }

    char* model;
    unsigned horsepower;
};
```

Методът void get\_model() const трябва да е връща char\* : char get\_model() const { return model; }. Методът void\* set\_model() const не трябва да е const и трябва да void: void set\_model() { delete[] this->model; this->model= new char(strlen(model)+1); strcpy(this->model,model); } В деструкторът не трябва да се трие unsigned horsepower, защото не е динамична памет.





## Въпрос 3

Частично правилен отговор  
12,25 от максимално 30,00 точки

Не е позволено използването на STL.

Спазвайте принципите на ООП. Придържайте се към добрите практики за писане на код.

## Tuple

Tuple ще наричаме наредена двойка от обекти, от различни типове (нека тези типове са T и S). Създайте клас Tuple, със следните методи:

- Конструктор по подразбиране
- Конструктор с параметри
- Методи за достъп до стойностите на наредената двойка
- Метод swap, който за наредената двойка T first, S second връща нова наредена двойка S second, T first
- Оператори за сравнение == и != - два Tuple са равни, когато съответните им елементи са равни.

## TupleContainer

Даден е клас TupleContainer, който ще пази неограничен (до колкото стига паметта) брой инстанции от Tuple.

Създайте метод за добавяне на нов Tuple обект към колекцията. Създайте метод, който връща броя обекти в колекцията. Предефинирайте оператора [], който да връща обекта, съответстващ на даден индекс. Ако индекса е невалиден, да се хвърли изключение





равни, когато съответните им елементи са равни.

## TupleContainer

Даден е клас `TupleContainer`, който ще пази неограничен (до колкото стига паметта) брой инстанции от `Tuple`.

Създайте метод за добавяне на нов `Tuple` обект към колекцията. Създайте метод, който връща броя обекти в колекцията. Предефинирайте оператора `[]`, който да връща обекта, съответстващ на даден индекс. Ако индекса е невалиден, да се хвърли изключение `out_of_range`. Предефинирайте оператора `==`. Два обекта са равни, когато техните `Tuple` са равни. Създайте метод `swap`, който прилага метода `swap` върху всеки `Tuple` и връща нов `TupleContainer<S, T>`, съдържащ новите наредени двойки.



4-62599.zip



### Въпрос 4

Частично правилен отговор  
23,00 от максимално 30,00 точки

**Позволено е използването на STL.**

**Спазвайте принципите на ООП. Придържайте се към добрите практики за писане на код.**

Напишете клас `User`, описващ потребител, със следните характеристики:





## Въпрос 4

Частично правилен отговор  
23,00 от максимално 30,00 точки

Позволено е използването на STL.

Спазвайте принципите на ООП. Придържайте се към добрите практики за писане на код.

Напишете клас User, описващ потребител, със следните характеристики:

- потребителско име
- парола
- дневна печалба от показване на реклами към този потребител
- методи, връщащи стойностите на член-данните на класа
- метод, изчисляващ месечната печалба от този потребител (за улеснение, приемаме че един месец средно има 30.416 дни)

Напишете клас PaidUser, описващ потребител, който е закупил абонамент. Съдържа следните характеристики:

- потребителско име
- парола
- име на абонаментния план
- месечна такса
- методи, връщащи стойностите на член-данните на класа
- метод, изчисляващ месечната печалба от този потребител





## Преглед



Напишете клас `PaidUser`, описващ потребител, който е закупил абонамент. Съдържа следните характеристики:

- потребителско име
- парола
- име на абонаментния план
- месечна такса
- методи, връщащи стойностите на член-данните на класа
- метод, изчисляващ месечната печалба от този потребител

Напишете клас `Users`, със следните характеристики:

- съдържа в себе си горе-дефинираните потребители
- метод за добавяне на нов потребител към колекцията
- метод, който връща какъв % от потребителите са с абонамент
- метод, който връща месечните приходи от всички потребители
- метод, който връща потребител по подаден индекс



4-62599.zip



# КН 20/21 - II. Поток

Изпит по ООП

! Решението трябва да спазва ООП парадигмата. Решения, които не реализират нужните класове, йерархия, абстракция и полиморфизъм, няма да бъдат признавани;  
! Заделянето и освобождаването на паметта са ваша отговорност;  
! В решенията не може да се използва `std::string`;  
! `std::vector` може да се използва в решението;

Задача

Да се реализира нов език за обмен на данни - FJSON (fake json), който представлява колекция от различни двойки (pairs).

Всяка двойка се състои от наименование (key) и стойност (value).

Всяко наименование трябва да бъде низ с произволна дължина.

Различават се четири типа двойки в зависимост от типа на стойността:

Съответно могат да се образуват различават 4 различни типа двойки:

Със стойност низ (StringPair) - низът е с произволна дължина

Пример:

ключ: `stringId`

стойност: `Hello World`

Със стойност дата (DatePair)

Пример:

ключ: `First uni day`

стойност: дата, състояща се от ден месец и година

Със стойност време (TimePair)

Пример:

ключ: `Work day start`

стойност: време, състоящо се от час и минути(часът е от 0 до 23 часа)

Със стойност дата и време (DateTimePair)

Пример:

ключ: `OOP Exam`

стойност: дата и време, състои се от ден, месец, година, час и минути

Представянето на стойността трябва да е оптимално за съответния тип двойка.

Коректността на входа за датата и времето НЕ е гарантирана. При валидацията за датата може да приемете, че дните във всички месеци са 31. При подадени невалидни стойности, същите се заменят с валидна стойност по подразбиране - 31 за ден, 12 за месец, 00:00 за време и се извежда подходящо съобщение.

Да се реализира член-функция `toString`, която да връща FJSON под формата на низ. Всяка двойка трябва да бъде форматирана в изходния низ по следния начин:

Със стойност низ (`StringPair`) - `{stringId : "Hello world"}`

Със стойност дата (`DatePair`) - `{First uni day : 15.10.2021}`

Със стойност време (`TimePair`) - `{Work day start, 10:15}`

Със стойност дата и време (`DateTimePair`) - `{OOP Exam: 23.06.2021, 09:04}`

Резултатния низ трябва да започва с '[' , да завършва с ']' и да са изредени неговите стойности, разделени със запетаи.

Пример:

```
{{stringId : "Hello world"}, {First uni day : 15.10.2021}, {Work day start, 10:15}, {OOP Exam: 23.06.2021, 09:04} ]
```

Реализирайте главна функция, или еквивалентни `unit tests`, която създава FJSON обект и множество стойности в него и тества реализираните функционалности.

## КН 19/20

### Задача за самостоятелна работа по ООП

Да се дефинира шаблон на клас `Vector<T>`, описващ колекция от последователни елементи от произволен тип `T`. Размерността на вектора е фиксирана при създаване и се задава в конструктора му. За класа да се реализират:

- операция `[]` за индексване, даваща достъп за четене и писане до елементите на вектора. Да се генерира подходяща грешка (напр. изключение) при опит за достъп до елемент с невалиден индекс.
- необходимите конструктори, деструктор, операция за присвояване
- оператори `+` и `+=`, събиращи векторите покомпонентно. Да се генерира подходяща грешка (напр. изключение), ако векторите са с различна размерност

Да се демонстрира употребата на класа с примерна програма, която дефинира чрез шаблона `Vector<T>`, матрица с размерност 2x3, представена като вектор от вектори от числа с плаваща запетая. Да се инициализират елементите на матрицата с нули.

За тази задача не се допуска използване на `std::vector` или други стандартни структури от данни от STL или предварително реализирани такива.

## Поправителен изпит по ООП

1. Да се дефинира клас `Word`, описващ дума, съставена от не повече от 20 символа от тип `char`. Класът да съдържа следните операции:

1. операция `[]` за получаване на *i*-тия пореден символ в думата
2. операции `+` и `+=` за добавяне на един символ в края на думата. Ако думата вече има 20 символа, операциите да нямат ефект
3. операции `<` и `==` за сравнение на думи спрямо лексикографската наредба
4. подходящи конструктори

2. Да се дефинира клас `Sentence`, описващ символен низ, състоящ се от произволен брой символи от тип `char`. Низът да се запазва в динамичната памет. Класът да поддържа следните операции

1. функция `addWord` за добавяне на дума (обект от клас `Word`) към края на изречението
2. класът да притежава всички необходими методи за осигуряване на правилен жизнен цикъл на обектите
3. метод за извеждане в изходен поток



3. Да се дефинира клас `EnglishSentence`, наследник на клас `Sentence`, който допуска изречения, съставени единствено от малки и големи латински букви и символите интервал, запетая, удивителен знак и точка.

Упътване: за целта да се дефинира подходящ вариант на метода `addWord`.

4. Да се дефинира функция

```
void addAndPrint ([подходящ тип] sentences, [подходящ тип] words[,...]),
```

където `sentences` е масив от произволен брой разнородни изречения, всяко от които може да е както обект от клас `Sentence`, така и обект от клас `EnglishSentence`, а `words` е масив от произволен брой думи. Ако е необходимо, функцията да има и допълнителни параметри. Функцията опитва да добави всяка от думите от `words` към всяко от изреченията в `sentences` чрез извикване на метода `addWord` и след това извежда всички така получени изречения.

За тази задача не се допуска използване на `std::vector` или други стандартни структури от данни от STL или предварително реализирани такива.

## Поправителен изпит по ООП Практикум

Unix time е формат за време, с който чрез едно единствено цяло число се представя момент във времето. Числото съответства на изтеклите секунди от 1 януари 1970 до дадения момент. Например, числото 1598618400 съответства на 15:40 часа на 28 август 2020 година.

1. Да се реализира абстрактен клас `WorkItem`, описващ задача със следните операции:

- `bool status(long now)` – връща `true`, ако задачата е приключила към момента `now` (в unix time формат) и `false` в противен случай
- `std::string name()` – описание на задачата

2. Да се дефинира клас `SingleTask`, наследник на `WorkItem`, който да дефинира задача със следните допълнителни характеристики, задавани по време на конструиране на задачата:

- Начален момент от време(unix time)
- Продължителност (в секунди)

3. Да се дефинира клас `TaskGroup`, наследник на `WorkItem`. Обектите от клас `TaskGroup` да поддържат списък с до 10 произволни задачи (прости и групови), които могат да принадлежат на повече от една група. Груповата задача се счита за приключена, когато всички нейни подзадачи са приключени. Класът да реализира следните допълнителни операции:

- `addTask` – добавяне на нова подзадача. В случай, че в груповата задача вече има максималния брой 10 задачи, функцията да връща `false`. За списъка с подзадачи можете да използвате `std::vector` или друга готова структура от данни
- `print` – извежда имената на всички подзадачи

Да се реализира примерна програма, която създава груповата задача, която съдържа в себе си както единични задачи, така и поне една друга груповата задача със собствени подзадачи.

Примерната програма да е съставена така, че всички заделени обекти да се унищожат коректно преди приключването ѝ.

# ИС 20/21

## Практикум

**Задачи, които не се компилират, силно нарушават принципите на ООП, имат memory leak, използват глобални променливи или използват библиотеките `string` и `vector`, ще бъдат автоматично оценени с 0т.**

**Чл. 163. (10) от Правилника на Софийския университет: “Студент, който при проверка на знанията въвежда в заблуждение чрез преписване, подсказване и др. п., се наказва по чл. 137, ал. 2 от Правилника на Софийския университет. Същото наказание се налага и при опит за подобно деяние.”**

## **Задача 1:**

Смолян е единствената област в България, в която няма жп транспорт. Бъдещото правителство иска да започне изграждане, за което обаче трябва да се измисли информационна система, която да не го свързва с останалите жп линии. Задачата ви е да реализирате система за жп линията. Естествено всяка гара трябва да има гари и влакове. Системата трябва да съдържа система от обвързани гари  
Гарата може да бъде 3 вида - крайна гара, разпределителна или обикновена гара.

Обикновената гара съдържа:

- име на гара - низ с максимум 20 символа
- сериен номер на гара - уникален идентификатор
- упътване към следващата гара по линията
- капацитет (брой влакове които може да бъдат на гарата по едно и също време. Броят е максимум 10)
- брой влакове на гарата
- конструктор, който приема име на гара, следващата гара, капацитет и правило по което се получава серийния номер, което използва капацитетът и името
- сигнал - показва дали може произволен влак да потегли
- метод, който позволява потеглянето на влаковете
- метод, който забранява потеглянето на влаковете
- метод, който спира гарата (забранява пристигането на нови влакове)

Крайната гара съдържа:

- име на гара - низ с максимум 20 символа
- сериен номер на гара - уникален идентификатор
- капацитет (брой влакове които може да бъдат на гарата по едно и също време. Броят е максимум 10)
- брой влакове на гарата

- конструктор, който приема име на гара, капацитет и правило по което се получава серийния номер, което използва капацитетът и името
- сигнал - показва дали може произволен влак да бъде изтеглен от мрежата (да бъде изваден)
- метод, който позволява изтеглянето на влак
- метод, който забранява изтеглянето на влак

Разпределителя съдържа:

- име на гара - низ с максимум 20 символа
- сериен номер на гара - уникален идентификатор
- разклонение към следващите 2 паралелни гари по линията
- капацитет (брой влакове които може да бъдат на гарата по едно и също време. Броят е максимум 10)
- брой влакове на гарата
- конструктор, който приема име на гара, следващата гара, капацитет и правило по което се получава серийния номер, което използва капацитетът и името
- сигнал 1 - показва дали може произволен влак да потегли по 1вото разклонение
- метод, който позволява потеглянето на влаковете по 1вото разклонение
- метод, който забранява потеглянето на влаковете по 1вото разклонение
- сигнал 2 - показва дали може произволен влак да потегли по 2рото разклонение
- метод, който позволява потеглянето на влаковете по 2рото разклонение
- метод, който забранява потеглянето на влаковете по 2рото разклонение
- метод, който спира гарата (забранява пристигането на нови влакове)

Всеки влак съдържа:

- номер на влак
- тип на влак (експресен - може да минава дори да е забранено минаването към следваща гара, бърз - може да преминава напред към следващата гара само ако е позволено, пътнически - може да преминава напред ако гарата е обикновена и е позволено и на

разпределител само ако е позволено да се мине и по двата разклона)

- индикатор показващ на коя гара се намира влакът
- конструктор с номер на влака, тип на влака (всеки влак започва от първата гара по линията)
- метод за преминаване на следваща гара по подаден номер на разклонение, ако е възможно (ако е невъзможно да се хвърля различна грешка в два случая, ако е забранено да се преминава заради сигнал или тип на влак, или ако не съществува разклонението)
- метод за изтегляне на влак, ако се намира на крайна гара и е възможно

Бонус: Да се реализира час на пристигане и заминаване на влак

## Задача 2:

Да се реализира шаблонен клас за стандартен моноид в C++, който съдържа единичен елемент от зададения тип; бинарна операция, която затваря моноида; функция на съгване - бинарна операция в n-мерния случай.

\* Да се напише метод `isIsomorphicTo`, която приема друг моноид и кандидат-функция за изоморфизъм и проверява дали двата моноида са изоморфни

\* Да се напише метод `constructInfiniteMonoidNaturals`, която конструира безкраен моноид, изоморфен на естествените числа

\* Да се напише метод за умножение на моноиди, в който единичните елементи се умножават, а бинарните операции се композират

**Забележка:** За да имаме изоморфизъм, задължително трябва да бъде първо хомоморфизъм. Това значи:

- $f(\text{единичен елемент}) = \text{единичен елемент}$
- $f(\text{елемент в моноида} \ (\text{бинарната операция в моноида}) \ \text{елемент в моноида}) = f(\text{елемент в моноида}) \ (\text{бинарната операция в моноида}) \ f(\text{елемент в моноида})$

Пример и подсказки: Направете си живота лесен и застопорете бинарната до единична и зациклете единичния елемент. Нека единичен елемент е 2 пък операцията е \*3. Тогава моноидът ще представлява безкрайна циклична група - 2, 6, 18, 54... Направете групата крайна - ограничете я, няма безкрайни оценки в този език. В най-честия случай операцията на съгване е просто многомерно прилагане на горната операция.

## Задача 3

Да се реализира имплементация на шаблонната структура Linked List, като след създаване на списъка, се правят произволен брой размествания на двойки кутии, стоящи на произволно генерирани "позиции" и да се провери дали новополучения списък е палиндром.

3->2->1->2->1

1,3 swap

2,4 swap

1->2->3->2->1

Изход: true

## Писмен изпит по ООП

**Решенията на всяко от подусловията да се демонстрират с подходящи примери!**

Файловата система на операционната система NoteBook поддържа единствено директории и кратки текстови бележки. Всяка директория може да съдържа произволен брой други директории и текстови бележки в себе си. Директориите, както и бележките, имат имена - произволни символни низове.

Устройство, работещо на NoteBook, може да се синхронизира с вашия компютър и да запише съдържанието на цялата си файлова система в един текстов файл.

1)[5 т.] Да се реализира абстрактен клас Entry, описващ елемент от файловата система на NoteBook. Entry да дефинира следните операции:

- `unsigned int size()`: размер на елемента
- `bool contains(const std::string& s)`: съдържа ли се низът `s` в елемента
- `Std::string name()`: име на елемента

2)[10 т.] Да се реализират наследник `Note` на `Entry`, описващ бележка, като:

- Размерът на бележката е броя на символите в нея
- `contains(s)` проверява дали `s` се среща като подниз в бележката

*Упътване: методът `find(std::string& substr)` на `std::string` проверява дали `substr` е подниз на даден низ и връща първата позиция, на която се среща, или `-1`, ако не се среща*

3)[15 т.] Да се реализират наследник `Directory` на `Entry`, описващ директория, като:

- Размерът на директория е сумата на размерите на съдържащите се в нея елементи
- `contains(s)` проверява дали `s` се среща в кой да е от елементите ѝ
- директориите имат метод `addEntry([подходящ тип]entry)`, който добавя към директория елемент, който може да е друга директория или бележка

4)[15 т.] От стандартния вход се прочита символен низ `s`. Да се изведат имената и размерите на всички елементи на файловата система, които съдържат низа `s` като подниз. В списъка се включват както бележките, така и директориите.

5)[20 т.] Да се състави формат за сериализация на файловата система на `NoteBook` и да се реализира сериализация за `Entry`.

Даден е пример за такъв формат. Можете да разработите него или произволен друг по ваш избор.

Следната директорийна структура:

```
root [dir]
  |-messages [note]
  |   |-saved for later [note]
  |   |-from mom [dir]
  |       |-mom 1 [note]
  |       |-to mom [dir]
  |       |-my_reply [note]
  |-jokes [dir]
      |-programmers [note]
```

Е представена текстово по следния начин:



Dir root	Главна директория с име "root"
2	съдържа 2 елемента
Dir messages	Директория с име "messages"
3	съдържа 3 елемента
Note 17 saved for later	Бележка "saved for later" с дължина 17 символа
I arrived safely	Съдържание на "saved for later"
Dir from mom	Директория с име "from mom"
1	Съдържа 1 елемент
Note 31 mom1	Бележка "mom1" с дължина 31 символа
Did you have something to eat?	Съдържание, забележете, че има нов ред
Dir to mom	
1	
Note 4 my_reply	
Yes!	
Dir jokes	
1	
Note 40 programmers	
Two C++ programmers enter into a bar...	

6)[25 т.] Да се реализира подходящо управление на паметта за вложените обекти (виртуални конструктори за копиране)

7)[30 т.] Да се десериализира файлова система, записана в създадения от вас формат