**Задача 1**

**Създаване на алгоритъм за намиране на най-ефективен график за присъствия**

**ВАЖНО!:** В този файл само дописваме или коментираме точките. НЕ СЕ ТРИЕ НИЩО! На отделна среща ще решим кое да остане, промени или премахне изцяло. Моля, поставяйте коментарите в квадратни скоби [Пример], за четимост.

**1.     Кратко описание на задачата**

При налични часове за присъствие на практически занятия, се иска да се разпределят студентите по удобен за тях график, като разпределението да е оптимално.

**2.     Входни данни**

-          Свободни часове на студентите (различни за всеки отделен студент)

-          Часове за провеждане на занятията (2 интервала по 4 часа на ден, 5 дни в седмицата)

-          Максимален брой на студентите в едно занятие (4, но може да се промени)

**3.     Изходни данни**

-          График на присъствие на студентите

-          График на заетост на стаята и брой на студентите в нея

**4.     Изисквания към решението**

-          Не е необходимо да се използва програмен код

-          Всеки студентите трябва да има между 20-40 часа/седмица

-          Търси се оптимално решение (всеки студент е желателно да е бил поне 1 път на ден )

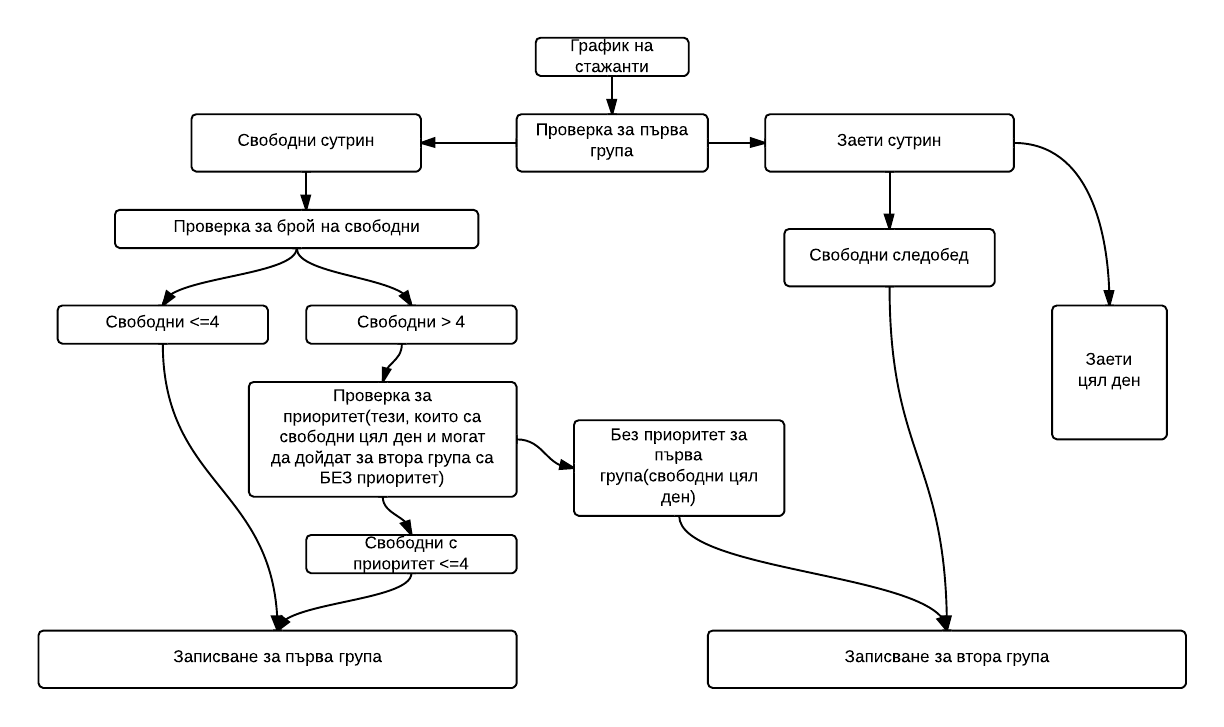
**5.     Изисквания към технологиите за реализиране на решението**

-          Да е приложимо на лист хартия, Word .docх файл или Excel spreadsheet

**6.     Кандидат алгоритми и готови чужди решения.**

[ По възможност, могат да се дават и примери за отделните алгоритми/решения ]

**a)** Както Пламен ни предложи, може да направим блок схема.А логиката на задачата си я представям по следния начин:

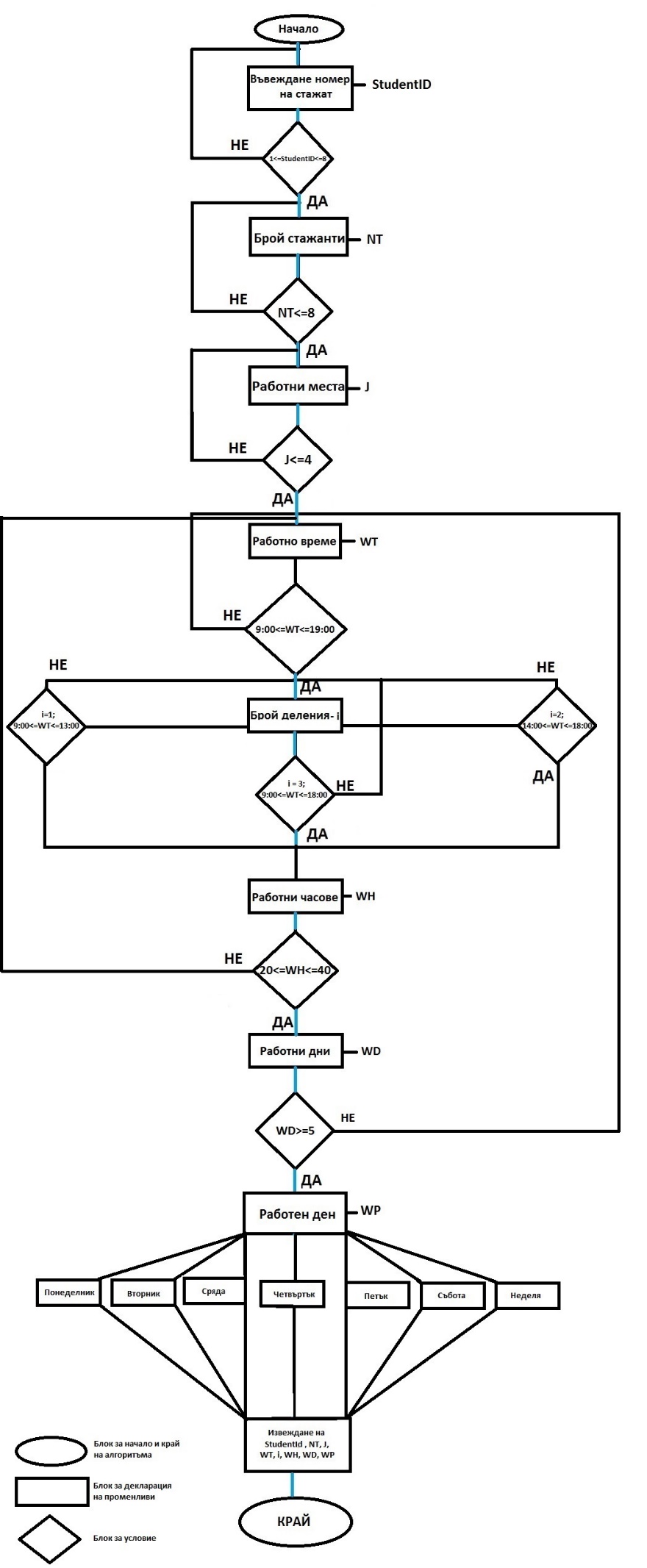
Започваме да проверяваме по азбучен ред на имената, кой е свободен за първа група, ако са <= 4 човека ги записваме. Ако са повече, проверяваме кои са свободни и следобед, като тези, които са свободни само сутрин ги записваме за първа група, а другите втора, т.е. тези с цял свободен ден са с по-нисък приоритет за първа група.

**б)** Въвеждане на графиците за всеки стажант. Въвеждане на параметри за изчисляването на работните часове: брой работни дни в седмицата, брой работни места, брой стажанти, допустимо работно време, брой деления в работното време, работни часове в едно деление, почивка между две деления. Определяне на точните времеви интервали за в рабоотните часове. За всеки ден от седмицата се избира стажантът с най-малко възможности за работно време и се вмъква в графика, след това се взима следващият по брой свободни часове и така нататък за всички стажанти и за всички работни дни, като в процеса на вмъкване на стажанти се прави проверка да няма повече от 40 работни часа и не по-малко от 20**.**

**в)** Направих блок-схема, която следва стъпките(до някаква степен) на Георги Георгиев, защото смятам, че правилно е описал самия подход на задачата. Имаме начало на блок схемата или някакъв вход. След като се въведе номера на стажанта се проверява дали има изобщо такъв стажант. Ако няма се връща отново на входа за ново въвеждане, докато номера на стажанта е правилен. След това се декларират броя на стажантите. Ако те са по-малко или равни на 8 се декларират работни места, ако не се връща обратно, докато броят на стажантите отговаря на условието. След като сме декларирали работните места трябва да удовлетворим друго условие, в което работните места да са по-малко или равно на 4. Ако това е ДА се преминава към следващата променлива(Работно време). Ако НЕ се изпълнява, докато условието стане вярно. Декларирано е работно време от 9:00 – 19:00. Ако студентът се запише в този интервал, се декларира следващата променлива (Брой деления). Ако обаче не въведе точно в този интервал, алгоритъма го връща докато влезне в същия.. Следващия етап са броя деления. В нашия случай 2(Първия от 9:00 – 13:00 и Втория от 14:00 – 18:00), но съм добавил и 3то деление, което е за 8 часа или цял работен ден. Стажанта си въвежда в кое делени(Смяна) ще работи даден ден и в последствие се пише колко работни часа е имал за месеца до момента. Ако има повече от 20 и по-малко или равно на 40 часа, алгоритъма влиза в следваща декларация, коята е Работни дни. Работните дни трябва да са повече или равни на 5. Ако НЕ са стажанта трябва да влезне в такова работно време, което да може да идва поне един път на ден. Ако това условие е изпълнено се избира в кой ден е работил съответно стажанта и се извеждат всички данни, които са удовлетворили условията. Алгоритъма приключва изпълнението си.

Може да уголемите мащаба на картинката, защото е  по-големичка.

                                    Владимир Метрашев



**Г)** Проблема е в основата си комбинаторен, търсейки най-добрата комбинация студент-час за група студенти и свободни часове.

Гледайки условията от т.2 – т.4 можем да заключим, че понеже дневно има  2 групи от по 4 студента, и всеки студент трябва да присъства 1 път на ден, следва че задачата се свежда до това в коя половина може да присъства всеки студент. Понеже това не ни устройва като адаптивност, приемаме че условията за максималния брой студенти не е строго 4, но ще влияе на оценката на резултата. Същото се отнася за броя присъствия на ден и правилото за 20-40 часа/сед., като имаме възможност за отработване между седмиците.

Понеже имаме 3 променливи (Графиците на студентите, графика на стаята и разписанието описващо комбинацията студент-час в стаята), подходите към проблема се явяват 3 (за всяка от променливите):

**- спрямо студента**

1. Преглеждайки на всички студенти графиците, определяме всичките им възможности за присъствие часове в седмицата с някакво условно означение (Пр. 0 – отсъства, 1 – присъства, 2 – през седмица, 3 - рядко).

2. Определяме тези, които покриват най-малко часове и ги записваме първи. При възникване на конфликт между повече от 4 хора с минимални часове (20), задачата няма решение.

3. След записа се преоценяват свободните места и се връщаме на стъпка 2 (понеже някой часове са се запълнили). Повтаря се процеса, докато няма хора с минимални часове.

4.  След това, по същия начин записваме останалите хора, имащи възможност да присъстват 20+4 часа. Тези с по-голям код имат по-голям приоритет (пр. 2 > 1).

5. Повтаряме точка 4 за хората с  20+8 часа, докато достигнем максималния брой часове (?? часа/седм). След това се извежда резултата.

**- спрямо местата:**

1 – 3 точка съвпадат със „спрямо студентите“.

4. Определяме часовете със най-малка посещаемост. Започва се от ранг 0 (никой не присъства): запълват се със студенти, като приоритета е 20+4, след което цифровия код на присъствието на студента. Ако студента има покрити максималния брой часове, се игнорира.

5. Повтаряме т.4 като повишаване ранга с 1. Спираме когато се свършат ранговете.

**- спрямо покритието на свободните часове на залата със свободните часове на студентите**

1 – 3 точка съвпадат със „спрямо студентите“.

4. Създават се комбинации от оставащите параметри от вида :

               ( студент, код на присъствие [>0] , записани до момента часове) – ( час, брой свободни места)

5. Прави се формула на оценка на тежеста на двойката (груба интерпретация):

               F =  код на присъствие / (бр. на други двойки с този студент + записани до момента часове +  брой свободни места + 1)

6. Взима се студента с най-голяма тежест и се прави преоценка на тежеста на двойките, където присъства студента или часа. При достигане на максимални часове за студента или бройки за часа, всички двойки в които те присъстват се премахват.

7. Процеса завършва, когато не останат повече двойки. Извежда се получения резултат.