**Динамични променливи на класа ІІ ч.**

**Композиция. Структуриране на проект в отделни модули**

*Задача:*

1. *Да се напише клас Fibonacci****,*** *който да притежава две целочислени член-променливи f1 и f2 и:*

* *Конструктор, инициализиращ променливите със стойност 1.*
* *Метод**Get(), който връща поредното число на Фибоначи, след числото – стойност на f2.*
* *Метод**Set(), който получава като параметър число и връща второто поредното число на Фибоначи, по-голямо от формалния параметър.*

1. *В класът Bank да се добави променлива FNumber от тип клас Fibonacci.*
2. *Да се актуализира type-conversion конструкторът на класа Bank, така че първоначално заделената памет да е число на Фибоначи, което е второто по-голямо от параметъра на конструктора.*
3. *Да се актуализират методите Append така, че ако се надхвърли размера на таблицата, да се задели памет, която е с размер – число на Фибоначи, по-голямо от предишния.*
4. *Проектът да се стуктурира така, че да се оформят 4 модула: за класа Fibonacci, за класа**ClientBank, за класа**Bank и за главната функция.*

***Упътване и указанния за работа:***

Преоразмеряването на динамичната памет чрез удвоявяне в класа *Bank* не е най-ефективния метод. Ако размерът на динамичната памет е число от редицата на Фибоначи се забелязва че паметта се увеличава с почти 2.6 пъти. Класът *Fibonacci* има за цел да определи размера на паметта, така че заделянето да бъде по-рационално. Методът *Set* на обекта *Fnumber* в класа *Bank* ще се използва, за да генерира размер на динамичната памет когато е необходимо.

Клас *Fibonacci*

|  |  |
| --- | --- |
| *int f1;*  *int f2;* | * *Fibonacci ();* * *int Get();* * *int**Set(int);* |

Клас *Bank*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *брой клиенти*  *размер на текущата памет*  *Fibonacci Fnumber;*  *......*   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |   *( динамичен масив)* | * *Конструктори* * *Метод, който добавя нов клиент* * *Метод, който премахва клиент* * ......... |

В проекта вече са разписани 1 структура и 3 класа. Заедно с дефенициите на техните методи правят кода претрупан и неподреден. По удачно е да се оформят отделни модули за всеки клас. Всеки модул се състои от два файла: с разширение *.h* и *.cpp*, които имат едно и също име. Хедърния файл ще съдържа само декларацията на класа, а дефинициите на неговите методи ще се намират в съответния .cpp файл. Връзката между отделните модули се реализира чрез #include на всеки хедърен файл в модула, където ще се ползва декларирания клас.