I. Description of the project

1. Overview of the project

The project aims to store all the necessary information about a fitness club that would allow for an accurate calculation of its budget, monthly profit (or loss), as well as provide tools to examine the components which account for the monthly change of the budget. The information stored should be fully modifiable to allow for the accurate portrayal of the current state of the fitness club. The user should be able to add, remove, modify and check the employees of the club, its clients as well as the relation between them.

2. Class and data structures overview

Public methods should resemble functionalities of an interface and not contain the helper functions. Test cases are also needed for the preliminary project

Classes:

- FitnessClub
- Client
- Employee
- List (template class

Class FitnessClub:

Attributes:

- **char* club_name** - a private array of chars storing the name of the club f.e. "Zdrofit".
- **char* address** - a private array of chars storing the address of the club.
- **List<Client>* clients** - a private object of the List template class with the specified type of class Client, it stores pointers to objects of class Client representing the clients of the fitness club.
- **List<Employee>* employees** - a private object of the List template class with the specified type of class Employee, it stores pointers to objects of class Employee representing the employees working in the fitness club.
- **double budget** - a private double representing the budget of the club. It can be negative as to signify debt of the fitness club.
- **double rent** - a private double representing the rent paid by the club. In the case where the club owns its own space it should be set to 0. Cannot be negative.

Based on the above attributes the class should be able to:

- create an object of class FitnessClub with and without parameters signifying the club name, address budget and rent
- have a destructor which prevents memory leakage
- set its attributes according to the users current needs
- add a change to the budget in the form of incrementing/decrementing it
- provide information on clients and employees already registered in the club's database

- provide information on clients with no employee assigned to them and vice versa
- create and insert new clients and employees into the database based on the information given by the user if it is correct and the list is not full
- assign client with the id given by the user to the employee given by the user if both exists
- unassign client with the id given by the user to the employee given by the user if both exists
- automatically assign all clients without an employee assigned to them to employees with the least amount of clients assigned to them
- remove clients and employees from their appropriate list safely i.e. without damaging data
- print financial information about the club containing current budget, rent to be paid monthly, the sum of salaries of all employees employed by the club, the sum of fees paid to the club by its clients and the club's monthly net profit (or loss if negative)
- provide information on the "worth" of each employee i.e. the sum of fees paid to the club by the clients trained by the employee
- change the attributes of clients and employees present on the fitness club clients and employees lists according to the data given by the user, as long as it abides by the rules imposed on each attribute
- change the max possible amounts of clients/employees that can be present in the fitness club database lists according to the data given by the user, as long as the new max amount is greater than the current amount of clients/employees on the list and not negative

Class Client:

Attributes:

- **char* name** - a private array of chars storing the name of the client.
- **char* surname** - a private array of chars storing the surname of the client.
- **long citizenID** - a private long storing the national identification number of the client for the purposes of the project it is assumed that the club operates in a country where such a number consists of 9 digits and no two citizen can possess the same number i.e. no two clients can have the same citizenID
- **double fee** - a private double representing the fee paid by the client monthly to the club for the provided services, cannot be negative
- **bool status** - a private boolean representing whether the client is currently trained by an employee of the fitness club or not (a client can be trained only by one employee)

Based on the above attributes the class should be able to:

- create an object of class Client with and without given parameters
- have a destructor which prevents memory leakage
- be able to copy the attributes of one client and copy it to another client
- return a reference to all attributes of the object
- set all the attributes of the object according to given parameters, as long as the fulfill the rules of chosen attribute

- use an overloaded << operator method to print out the information on the attributes of the object

Class Employee:

Attributes:

- **char* name** - a private array of chars storing the name of the employee.
- **char* surname** - a private array of chars storing the surname of the employee.
- **long citizenID** - a private long storing the national identification number of the employee for the purposes of the project it is assumed that the club operates in a country where such a number consists of 9 digits and no two citizen can possess the same number i.e. no two employees can have the same citizenID
- **double salary**- a private double representing the fee paid by the client monthly to the club for the provided services, cannot be negative
- **List<Client>* clients** - a private object of the List template class with the specified type of class Client, it stores pointers to the Client objects representing clients trained by the chosen employee

Based on the above attributes the class should be able to:

- create an object of class Employee with and without the given parameters
- have a destructor which prevents memory leakage
- be able to copy the attributes of one client and copy it to another client
- return a reference to all attributes of the object
- set all the attributes of the object according to given parameters, as long as the fulfill the rules of chosen attribute
- add clients to the List<Client>* clients attribute and giving them a node id specified in the parameter given along with the Client object
- remove clients from its list without deleting the objects pointed to by the pointers stored in the removed nodes of the list
- use an overloaded << operator method to print out the information on the attributes of the object

Class List

Attributes:

- **struct Node** - a private structure definition which defines a list node with the following attributes:
  - **T* data** - a data attribute of the Node structure storing a pointer to an object of the specified type T
  - **Node* next** - a pointer to the next Node of the list. Can be nullptr, which signifies the end of the list.

    - **long id** - a long storing the Id of the node. Used to distinguish objects on the list. Cannot be less than 1 and should be given based on the class attribute static int id or an explicitly given id number in the insertNode method.

- **Node* head** - a private pointer to the head node of the list
- **long amount** - a private long storing the amount of nodes currently present on the list. Cannot be less than 0 or larger than max_amount attribute
- **long max_amount** - a private long representing the maximum amount of nodes which can be in the list. Cannot be set to 0 or less.
- **long id** - a private long storing the id number of the list, when an empty list is created it should be set to 0. Each time a new node is inserted with insertNode it should be incremented by 1 and its value should be given to the id attribute of the newly created node.

Based on the above attributes the class should be able to:

- create an object of class List with a specified type
- have a destructor which prevents memory leakage
- print a list of all objects of class <T> stored in the object of class LIst<T>
- change the max_amount attribute
- give information on the amount of objects currently stored on the list, the max amount of objects which can be stored on the list, the id attribute and information whether list is empty
- find a node with the given id and return the pointer to data stored by it
- iterate to nth object on the list and return a reference to the object pointed by the pointer stored in the data attribute of the node
- iterate to nth object on the list and return the list id of that node
- remove all nodes stored on the list

3. Restrictions, limits, assumptions

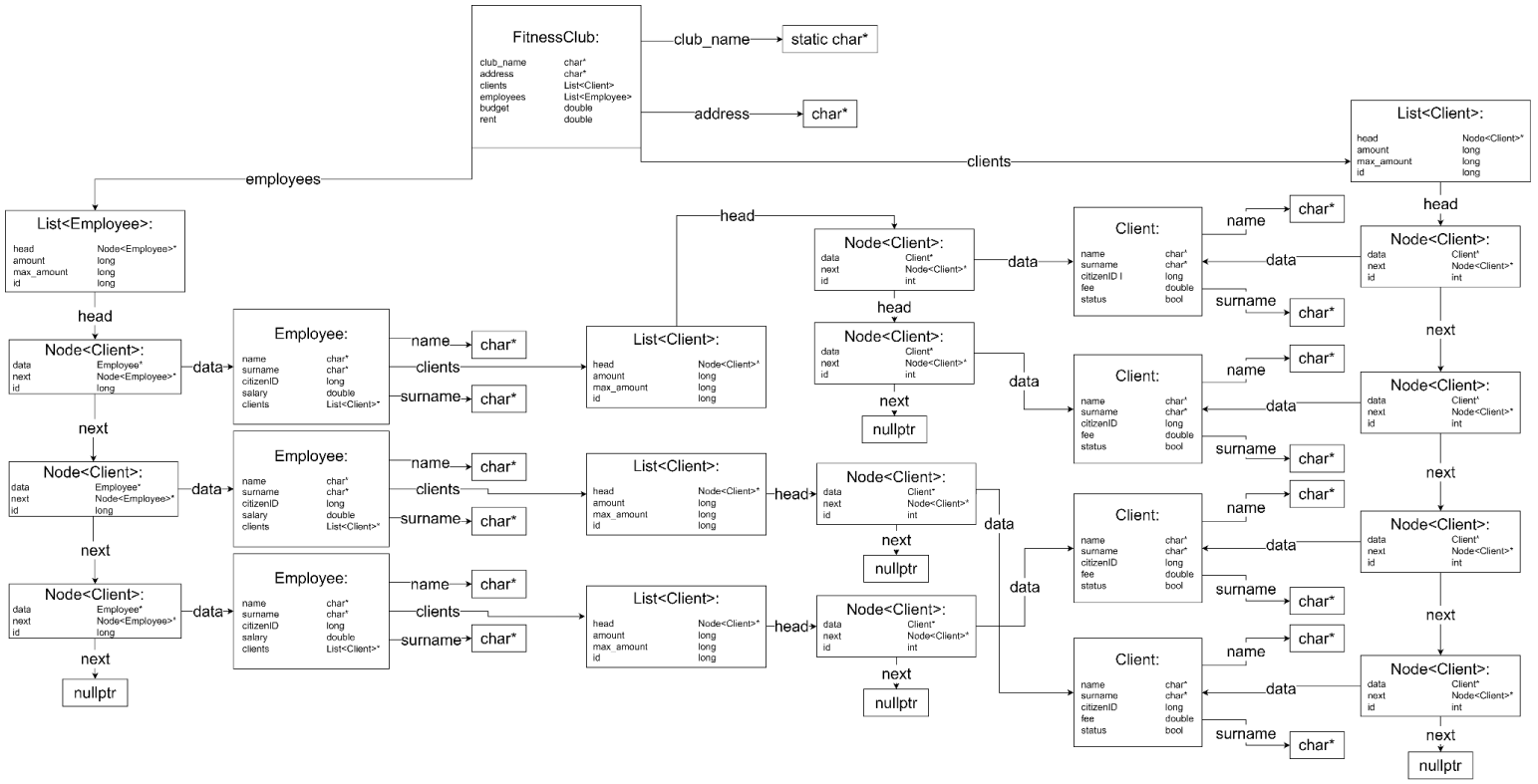
The program assumes the following:

- everyone who is eligible to be registered as a client or employee has a 9-digit unique nation identification number
- no employee will ever need to have more than 500 clients in his clients list attribute, thus no method for changing the max amount of clients on employee client list is provided

During the normal operation of the program the following limits and restrictions should be implemented:

1. FitnessClub attribute rent must be greater or equal to 0.
2. FitnessClub attribute budget can be negative.
3. No two clients can have the same citizenID.
4. No two employees can have the same citizenID.
5. An employee can also be a client of the club, in this case a client can have the same citizenID as an employee.
6. A client can be printed only if the given id belongs to a client on the clients list, else a warning should be printed.
7. An employee can be printed only if the given id belongs to an employee on the employees list, else a warning should be printed.
8. An attempt to assign a client to an employee in the case where either the employee or the client (or both) with the given ids do not exist, should fail and print a warning.
9. An attempt to unassign a client from an employee in the case where either the employee or the client (or both) with the given ids do not exist, should fail and print a warning.
10. An attempt to remove a client or employee in the case where no client/employee has the given id, should fail and print a warning.
11. An attempt to change an attribute of a client or employee in the case where no client/employee has the given id, should fail and print a warning.
12. In the case of changing the citizenID attribute of either a client or an employee the method should also check if another client/employee doesn't already have that citizen ID number.
13. The fee paid by a client or a salary paid to the employee both cannot be negative, thus when given a negative value the chosen method should transform it into a positive one before assigning it to the attribute.
14. The maximum amount of objects on the list is modifiable and can be set by the user, by default the limit is set to 500, if the amount of the objects on the list is equal to max amount the attempt to insert a new object will not succeed and a warning will be printed.
15. Two FitnessClub objects should be able to exist at the same time and function correctly.
16. A client cannot train himself i.e. if a client and an employee are the same person (same citizen id) then the client object cannot be assigned to the employee object.

II. Case study (a memory map)



[Link to the diagram as it might be difficult to see for some](#)