**Report**

*Yernur Nurambek*

*Ilyas Mukhlis*

*Tileuzhan Mukhamet*

*Sultan Kobeyev*

**Algorithm**

1. To allocate first the most problematic courses, function called lectureSort() sorts the courses based on the number of groups that should take the course. The courses with the greatest number of groups are placed in the first positions of the array of objects courses[]. After this, labs and tutorials are placed.
2. The algorithm starts allocation with the first course in array of objects courses[]:
   1. The program calls function randAlloc(), which chooses a random day and timeslot, then checks whether groups and instructor which participate in this course are available at this time.
   2. If they are available then it chooses a room available at this time with the required capacity by calling function findAvailableRoom().
   3. If room is found program sets time and room for this course and goes to next course for random allocation.
   4. If room is not found or the groups and instructors are busy at this random time the program calls function orderAlloc() which checks all days and time slots and chooses the first one for which all conditions (groups, instructors and room are available) are satisfied.
   5. If such timeslot is found program calls function randAlloc() to allocate randomly next course.
   6. If no timeslot is found to satisfy the imposed conditions, then error is displayed, and randAlloc() function is called to continue random allocation of next course. In this case the course is not allocated.

**Algorithm advantages:**

1. Generates different timetable in each execution.
2. Efficiently allocates rooms: big rooms are preserved for courses with larger number of participants.
3. Starts with the most problematic courses.
4. Does not go to infinite loop.
5. Simple therefore quick
6. Can be easily changed/modified without having to rewrite the entire program.

**Sample input**

Sample input file consists of CoursesList.txt, InstructorsList.txt, RoomList.txt, GroupList.txt.

Output can be viewed in TimeTable.txt.

**The main difficulties that were encountered during writing the program:**

* **Choosing an algorithm to generate a timetable.**

At the initial stages of the project, many different algorithms that are usually used in generating timetables (such as Backtracking, Genethic Algorithm and Tabu Search) were considered. However, because of the limited programming knowledge that we had at that time, these algorithms were difficult to apply for our project. After considering many algorithms it was decided to create a simple algorithm, which will allocate all courses in timeslots by checking the limiting conditions.

* **Choosing the data structure for the program.**

Initially, it was decided to use multidimensional arrays to store all the information and timetables for courses, rooms, instructors and groups. Since the arrays can store data of only one type, it was difficult to link different items (e.g. course name, course size, days, timeslots etc.) in one array. Then, it was realized that it is possible to create array of objects, in which each element can have its own set of attributes that can be of different types.

* **Difficulties with input file.**

Initially, the input file was written so that the program reads only one word in each line, therefore the input file became quite long and was difficult to manage. This problem was overcome by inserting the comma after each word. In this way, the function that takes input has to check only until subsequent comma, which made it possible to write all information regarding one object (course, instructor, group) in one line and facilitated managing the input file.

* **Accessing the information stored inside object of one class by object of another class.**

When allocating a particular course in timeslot and room, the program has to check if group that attends this course and instructor are available. To check their availability the program has to find the object in the arrays groups[], instructors[], where this information stored. To avoid searching the necessary object during each course allocation, the attributes for indexes (the element number of array) for groups, instructor and rooms were declared and function indexDistribution() waswritten. indexDistribution() finds the instructor that teaches a certain course and groups that should attend it and assigns the index of cell in array of objects were their data are stored.

* **Optimization:**

To optimize program index system was introduced.

In algorithm to check whether group or instructors are free the program should check group and instructor timetable. For this purpose it should know the position of instructor or group in array of objects of instructors and groups. This would require a function which search group or instructor position in array of objects. Instead of search each time for this position this is done only once at the beginning of the program after input. And all indexes are stored in class course, this significantly decreases number of calculations which program performs.