Capstone Project (IT205)

Algo Warriors

Section: A (P1)

AutoMatic TimeTable Generator

Team Members:

Parth Bhatt (202301022)

Heet Bhuva (202301151)

Om Patel (202301027)

Deep Kakadiya (202301097)

Abstract:

This C++ code implements a program for generating and managing academic timetables based on input data including course details, professor information, and semester schedules. The program organizes courses into time slots across different days of the week and rooms, ensuring professors' availability, credit allocation for courses, and adherence to branch and semester constraints. It uses file input/output operations to read course information from external files and writes the generated timetable to CSV files. Additionally, the program provides functionalities to generate timetables for specific professors or semesters and also allocates Elective courses, facilitating efficient scheduling and management within an educational institution. The core functionality revolves around allocating courses to available time slots while considering various constraints, such as professor availability, credit requirements, and branch-specific arrangements, thus automating the process of timetable generation in an academic setting.

In this Automatic Timetable generator we are using Array data structure and Hash table data structure. Overall Space Complexity of the Code is O(n + Total_Rooms * Total_slots * 5) while the Overall time Complexity of the code is O(p*r*n). Although we are using many names, most of them are constant values like Total Rooms, Total Slots and Number of Days in Week so the time complexity of this code is very less. Less time complexity and Space complexity is one of the reasons for using array data structure. Arrays are simple and straightforward data structures. They provide direct access to elements based on indices, which can be efficient for certain operations such as accessing elements by position. In scenarios where the data structure's requirements are minimal and straightforward, arrays can be more efficient than more complex data structures.

n = Total number of Courses
p = Total courses taught by specific Professor
r = Total rooms
m = length of string

Note: We are considering that B.Tech ICT students are in Section-A and B.Tech ICT+CS, MnC and EVD students are in Section-B. For Core courses, timings are 8:00 AM to 1:00 PM. Also for Elective Courses, we are taking a time slot from 4:00 PM to 6:00 PM. Full names of all professors are mapped in the main function which can be accessed by Professor code.

Step by Step Explanation:

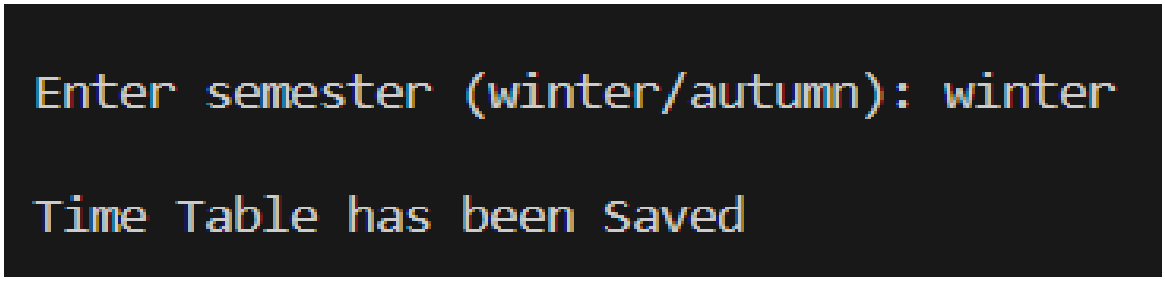
1. Firstly,we define vectors to store various things like semester of course,professor name for each course,frequency of course in a week of each course and branch of each course.

2. Then we ask the user for which semester he wants to see the time table,if he print winter then we take input from the **“winter.txt”** file in our vectors which I mentioned in first point and he print autumn then we take input from **“autumn.txt”** file in our vectors.

3. Then we call our “**Generate_Timetable()”** function which stores course code in a 3-D vector called **“Course_Allocation[room][slot][day]”** by checking whether there is space to allocate class in a specific room for a specific day and time.

4. **“Generate_Timetable()”** function calls an important function called **“Is_Prof_Avail”** which checks whether the professor of that course have an another class at that day and time or not.If the professor is free at that time slot this function return true and given course will assign to that professor and if the professor is not free,it will return false so that the given course will assign on another day or time slot.

5. Now,the only thing left is to print TimeTable on screen.It will be done by **“WriteTimetableToFile(string& filename)”** function. Our TimeTable will be downloaded on your laptop when you select either of the semester winter or autumn.If you select winter then winter timetable will downloaded and same for autumn.The downloaded file name will be **“timetable.csv”**.It has 7 columns which are for displaying day,time,room,course,professor,semester and branch respectively for each course.



timetable

Day	Time	Room	Course	Professor	Semester	Branch
Monday	8:00 AM - 9:00 AM	LT1	HM106(A)	BK	2	B.Tech ICT
		LT2	IT205(B)	AR2	2	B.Tech ICT+CS; EVD
		LT3	HM116(A)	GAURAV	4	B.Tech ICT
		CEP-003	HM116(B)	CYRIL	4	B.Tech ICT+CS
		CEP-103	MC226	RG	4	B.Tech MnC
		CEP-104	SC407(B)	YV	6	B.Tech ICT+CS
		CEP-105	IT585	RC	2	M.Tech ICT
		CEP-106	SC602	NKS	2	M.Sc DS
		CEP-107	PC731	VP	2	M.Des CD
	9:00 AM - 10:00 AM	LT1	IC121(A)	SR	2	B.Tech ICT
		LT3	EL203(A)	BM	4	B.Tech ICT
		CEP-003	IT214(B)	PMJ	4	B.Tech ICT+CS
		CEP-104	CS302(B)	AR	6	B.Tech ICT+CS
		CEP-105	CT421	HP	2	M.Tech EC
		CEP-106	IT608	SB	2	M.Sc DS
		CEP-107	PC750	ADUTTAGUPTA	2	M.Des CD
	10:00 AM - 11:00 AM	LT1	SC205(A)	AT2	2	B.Tech ICT
		LT2	HM106(B)	BK	2	B.Tech ICT+CS; MNC; EVD
		LT3	CT216(A)	YV	4	B.Tech ICT
		CEP-102	MC221	GP	4	B.Tech MnC
		CEP-105	IT620	TBD2	2	M.Sc IT
		CEP-106	MA622	IIRS	2	M.Sc AA
	11:00 AM - 12:00 PM	LT1	SC217(A)	GD	2	B.Tech ICT
		LT2	MC215	SB2	2	B.Tech MnC
		CEP-102	MC223	PB	4	B.Tech MnC
		CEP-104	CS301(B)	BC	6	B.Tech ICT+CS
		CEP-105	IT694	SS	2	M.Sc IT

(Screenshot of Output)

Additional Features:

1. We added a feature in which if you want TimeTable with respect to professor we can provide it.It will done by a function called **“Professor_TimeTable(const**

string& professorName)".We asked the user if he want timetable with respect to professor.If the user enter YES,we will ask for professor code and as the user enter the professor code,the file of timetable for that professor will be downloaded on user laptop. The downloaded file name will be **"professorName+_timetable.csv"**.It has 6 columns which are day, time, course, semester, branch and room. Let's see some of the screenshots for this feature.

```
Do you want time table with respect to Professor? (YES/NO)
YES

Enter the Professor code for whome you need Timetable.
YV

Time table for YV has been saved successfully
```

Yash Vasavada_timetable

Day	Timing	Course	Semester	Branch
Monday	8:00 AM - 9:00 AM	SC407(B)	6	B.Tech ICT+CS
	10:00 AM - 11:00 AM	CT216(A)	4	B.Tech ICT
	12:00 PM - 1:00 PM	SC407(B)	6	B.Tech ICT+CS
Tuesday	8:00 AM - 9:00 AM	SC407(A)	6	B.Tech ICT
	10:00 AM - 11:00 AM	CT216(A)	4	B.Tech ICT
	12:00 PM - 1:00 PM	SC407(B)	6	B.Tech ICT+CS
Wednesday	8:00 AM - 9:00 AM	SC407(A)	6	B.Tech ICT
	11:00 AM - 12:00 PM	CT216(B)	4	B.Tech ICT+CS
Thursday	8:00 AM - 9:00 AM	SC407(A)	6	B.Tech ICT
	11:00 AM - 12:00 PM	CT216(B)	4	B.Tech ICT+CS
Friday	9:00 AM - 10:00 AM	CT216(A)	4	B.Tech ICT
	11:00 AM - 12:00 PM	CT216(B)	4	B.Tech ICT+CS

2. We also added a feature in which if you want TimeTable for a particular semester, we can provide it.It will done by a function called **“Semester_TimeTable(const string& semester)”**.We asked the user if he want timetable for a particular semester.If the user enter YES,we will ask for semester number and as the user enter the semester,the file of timetable for that semester will be downloaded on user laptop. The downloaded file name will be **“semester+_timetable.csv”**.It has 6 columns which are day, time, course, professor, branch and room.

2_timetable

Day	Time	Course	Professor	Branch	Room	
Monday	8:00 AM - 9:00 AM	HM106(A)	BK	B.Tech ICT	LT1	
		IT205(B)	AR2	B.Tech ICT+CS; EVD	LT2	
		IT585	RC	M.Tech ICT	CEP-105	
		SC602	NKS	M.Sc DS	CEP-106	
		PC731	VP	M.Des CD	CEP-107	
	9:00 AM - 10:00 AM	IC121(A)	SR	B.Tech ICT	LT1	
		CT421	HP	M.Tech EC	CEP-105	
		IT608	SB	M.Sc DS	CEP-106	
		PC750	ADUTTAGUPTA	M.Des CD	CEP-107	
	10:00 AM - 11:00 AM	SC205(A)	AT2	B.Tech ICT	LT1	
		HM106(B)	BK	B.Tech ICT+CS; MNC; EVD	LT2	
		IT620	TBD2	M.Sc IT	CEP-105	
		MA622	IIRS	M.Sc AA	CEP-106	
	11:00 AM - 12:00 PM	SC217(A)	GD	B.Tech ICT	LT1	
		MC215	SB2	B.Tech MnC	LT2	
		IT694	SS	M.Sc IT	CEP-105	
	12:00 PM - 1:00 PM	IC121(B)	PK	B.Tech ICT+CS; EVD	LT1	
		MC125	MS	B.Tech MnC	LT2	
		IT629	TBD3	M.Sc IT	CEP-105	
		MA624	IIRS	M.Sc AA	CEP-106	

3. Time Table for Elective courses are also made and will be saved by name “timetable_elective”.

```
Do you want Time Table for Elective courses? (YES/NO)

YES
Enter semester for Elective (winter/autumn): winter


Time table for elective has been saved.
```

timetable_elective

Day	Time	Room	Course	Professor	Semester	Branch
Monday	4:00 PM - 5:00 PM	CEP-003	IT422(E)	RM	4	B.Tech ICT Elective
				RM	6	B.Tech ICT Elective
				RM	8	B.Tech ICT Elective
		CEP-102	IE411(E)	AMM	6	B.Tech ICT Elective
		CEP-103	IE404(E)	MK	6	B.Tech ICT Elective
		CEP-104	CT491(E)	DKG	6	B.Tech ICT Elective
				DKG	8	B.Tech ICT Elective
		CEP-105	EL464(E)	SR	6	B.Tech ICT Elective
				SR	8	B.Tech ICT Elective
		CEP-106	EL497(E)	SK	6	B.Tech ICT Elective
				SK	8	B.Tech ICT Elective
		CEP-107	IT449(E)	PB	6	B.Tech ICT; MnC Elective
				PB	8	B.Tech ICT; MnC Elective
		CEP-108	IT412(E)	SDG	6	B.Tech ICT Elective
				SDG	8	B.Tech ICT Elective
		CEP-109	IT495(E)	GP	6	B.Tech ICT; MnC Elective
				GP	8	B.Tech ICT; MnC Elective
		CEP-110	SC471(E)	MT	6	B.Tech ICT; MnC Elective
				MT	8	B.Tech ICT; MnC Elective
		CEP-202	HM327(E)	MM	6	B.Tech ICT Elective
				MM	8	B.Tech ICT Elective
		CEP-203	HM488(E)	AMISHALMODI	6	B.Tech ICT Elective
				AMISHALMODI	8	B.Tech ICT Elective
	5:00 AM - 6:00 AM	CEP-003	IE402(E)	MKR	4	B.Tech ICT Elective
		CEP-102	IE403(E)	PKS	6	B.Tech ICT; MnC Elective
		CEP-103	CT548(E)	MK2	6	B.Tech ICT Elective

Pseudo Code:

function main(){

```
  label:
     Print "Enter semester (winter/autumn): "
     input semester

     Initialise filename
     if (semester == “winter")
        filename = "winter.txt"
     else if (semester == “autumn")
        filename = "autumn.txt"
     else
        Print "Please Enter valid Input. Try again."
        goto label

     input = open file(filename)
     if (input is not open)
        Print "Error: Unable to open file " + filename
        return 1

     while (there are lines in input file)
     {
        perticularline = read line from input
        iss = initialize input string stream with perticularline
        sem, sub, pro_n, cre, br = "", "", "", "", “"

        if getline(iss, sem, ',') and getline(iss, sub, ',') and getline(iss, pro_n, ',') and getline(iss,
cre, ',') and getline(iss, br, ‘,')
        {
           push sem to semesterinput
           push sub to subjectinput
           push pro_n to professor_nameinput
           push cre to creditinput
           push br to branchinput
         }
       }

     Generate_Timetable()

     WriteTimetableToFile("timetable.csv")
     Print "Time Table has been Saved"

     Print "Do you want time table with respect to Professor? (YES/NO)"
```

```
    input yesno

    if (yesno == “YES”)
    {
       proff_name = ""
       Print "Enter the Professor code for whom you need Timetable."
       input proff_name
       Professor_TimeTable(proff_name)
       Print "Time table for " + proff_name + " has been saved successfully"
    }

    Print "Do you want Time Table according to semester? (YES/NO)"
    input yn

    Semester = ""
    if (yn equals “YES”)
    {
       Print "Enter Semester Number."
       input Semester
       Semester_TimeTable(Semester)
       output "Time table for " + Semester + " semester has been saved successfully"
     }
    return 0

}



function string_match(branch_index, k){

  branch_index2 = empty vector of strings
  curr_branch = empty vector of strings

   push branchinput[k] to curr_branch
   push branchinput[branch_index] to branch_index2

  found = (curr_branch[0].find(branch_index2[0]) != npos)

  if(found)
      return true
   else{
     found = (branch_index2[0].find(curr_branch[0]) != npos)
     if(found)
         return true
      else
        return false
```

```
    }
}
```

```
function Is_Prof_Avail(timing, day, Prof, room, branch_index){

  code = empty vector of strings

  for(I from 0 to length of professor_nameinput - 1)
  {
      if Prof is equal to professor_nameinput[I]{
          push subjectinput[i] to code
      }
  }

  if(timing < 4)
  {
      if(timing > 0)
      {
        for(t from timing - 1 to timing + 1)
        {
            for(j from 0 to length of code - 1)
            {
                for(I from 0 to room)
                {
                    if Course_Allocation[i][t][day] is equal to code[j]:
                        return false
                }
             }
         }
       }
      }
   else
   {
        for(t from timing to timing + 1)
        {
            for(j from 0 to length of code - 1)
            {
                for(I from 0 to room)
                {
                    if(Course_Allocation[I][t][day] == code[j])
                        return false
                 }
              }
```

```
            }
          }
        }

    else
    {
      for(t from timing - 1 to timing)
      {
          for(j from 0 to length of code - 1)
          {
              for(i from 0 to room)
              {
                  if (Course_Allocation[l][t][day] == code[j])
                      return false
              }
           }
      }
     }

  for (i from 0 to room - 1)
  {
      for (k from 0 to length of subjectinput - 1)
      {
          if (Course_Allocation[l][timing][day] == subjectinput[k])
          {
              if (branchinput[k] == branchinput[branch_index]) or (string_match(branch_index, k)))
              {
                  if (semesterinput[k] == semesterinput[branch_index])
                      return false
               }
           }
       }
     }

    return true

}
```

function Generate_Timetable(){

```
  course_index = 0

  while (course_index < length of subjectinput)
```

```
{
    for (i from 0 to Total_Rooms - 1)
    {
        for (j from 0 to Total_slots - 1)
        {
            for (k from 0 to 4)
            {
                if (Course_Allocation[l][j][k] == "0" and course_index < length of subjectinput and
ls_Prof_Avail(j, k, professor_nameinput[course_index], i, course_index) and
stoi(creditinput[course_index]) > 0)
                {
                  Course_Allocation[i][j][k] = subjectinput[course_index]

                 creditinput[course_index] = to_string(stoi(creditinput[course_index]) - 1)

                 if (stoi(creditinput[course_index]) == 0)
                     course_index++
                }
              }
            }
         }
     }
```

function Professor_TimeTable(professorName){

```
  filename = concatenate(shortToFullName[professorName], "_timetable.csv")
  outfile = open(filename)

  if(outfile does not exist){
     print "Error: Unable to open file"
  }

 write "Day,Time,Course,Semester,Branch,Room" to outfile

 for(f from 0 to length of professor_nameinput)
 {

    if(professor_nameinput[f] == professorName)
    {

        for(l from 0 to Total_Rooms)
        {
```

```
            for(j from 0 to Total_slots)
            {

               for(k from 0 to 4)
               {

                  if (subjectinput[f] == Course_Allocation[l][j][k])
                  {

                     if k == 0:
                        write "Monday," to outfile
                     else if k == 1:
                        write "Tuesday," to outfile
                     else if k == 2:
                        write "Wednesday," to outfile
                     else if k == 3:
                        write "Thursday," to outfile
                     else if k == 4:
                        write "Friday," to outfile

                     if j == 0:
                        write "8:00 AM - 9:00 AM," to outfile
                     else if j == 1:
                        write "9:00 AM - 10:00 AM," to outfile
                     else if j == 2:
                        write "10:00 AM - 11:00 AM," to outfile
                     else if j == 3:
                        write "11:00 AM - 12:00 PM," to outfile
                     else if j == 4:
                        write "12:00 PM - 1:00 PM," to outfile

                     write subjectinput[f] + "," + semesterinput[f] + "," + branch[branchinput[f]] + “,"
to outfile

                     if i == 0:
                        write "LT1," to outfile
                     else if i == 1:
                        write "LT2," to outfile
                     else if i == 2:
                        write "LT3," to outfile
                     else:
                        write "CEP-" + (i + 1) + "," to outfile
                  }
               }
            }
         }
```

```
    }
  }


  close outfile

}
```

```
function WriteTimetableToFile(filename){
   outfile = open(filename)

   if(outfile does not exist)
      print "Error: Unable to open file"


   # Write headers to the file
   write "Day,Time,Room,Course,Professor,Semester,Branch" to outfile

  days = ["Monday", "Tuesday", "Wednesday", "Thursday", “Friday"]

  times = ["8:00 AM - 9:00 AM", "9:00 AM - 10:00 AM", "10:00 AM - 11:00 AM", "11:00 AM -
12:00 PM", "12:00 PM - 1:00 PM"]

   # Initialize arrays to keep track of whether each day and time has been printed
   printedDays = [false, false, false, false, false]
   printedTimes = [false, false, false, false, false]

  for(k from 0 to 4)
  {
      day = days[k]

     if not printedDays[k]:
         dayPrinted = false

        for(j from 0 to 4)
        {
            time = times[j]
            timePrinted = false

            slotPrinted = false

           for(l from 0 to 22)
```

```
{

    if(Course_Allocation[l][j][k] != “0”)
    {

        if not dayPrinted
        {
            write day to outfile
            dayPrinted = true
            printedDays[k] = true
        }
        else{
            write "," to outfile
        }

        if not timePrinted
        {
            write time to outfile
            timePrinted = true
            printedTimes[j] = true
        }
         else{
            write "," to outfile
        }

        if i == 0:
            write "LT1" to outfile
         else if i == 1:
            write "LT2" to outfile
         # Continue writing room names for other indices.
         else:
            write "CEP-" + (i + 1) to outfile

         write  Course_Allocation[i][j][k]  to outfile

        courseIndex = -1

         for(idx from 0 to length of subjectinput)
         {
            if(Course_Allocation[l][j][k] == subjectinput[idx])
            {
               courseIndex = idx
               break
            }
          }
```

```
                    if(courseIndex != -1)
                    {
                        write professor_nameinput[courseIndex] + "," + semesterinput[courseIndex] +
"," + branchinput[courseIndex] to outfile
                    }

                    write newline to outfile
                    slotPrinted = true
                }

                if(not slotPrinted and dayPrinted)
                    write  time + newline to outfile
            }
        }

  close outfile
}
```

function Semester_TimeTable(Semester){

```
  outfile = open(filename)

   if(outfile does not exist)
   {
      print "Error: Unable to open file", filename
   }

   # Write headers to the file
   write "Day,Time,Course,Professor,Branch,Room" to outfile

  days = ["Monday", "Tuesday", "Wednesday", "Thursday", “Friday"]

   times = ["8:00 AM - 9:00 AM", "9:00 AM - 10:00 AM", "10:00 AM - 11:00 AM", "11:00 AM -
12:00 PM", "12:00 PM - 1:00 PM"]

   # Initialize arrays to keep track of whether each day and time has been printed
   printedDays = [false, false, false, false, false]
   printedTimes = [false, false, false, false, false]

  for(k from 0 to 4)
  {
      day = days[k]
```

```
if(not printedDays[k])
{
    dayPrinted = false

   for(j from 0 to Total_slots - 1)
   {
       time = times[j]
       timePrinted = false

       slotPrinted = false

      for(l from 0 to Total_Rooms - 1)
      {

         for(f from 0 to length of professor_nameinput - 1)
         {

            if(semesterinput[f] == Semester and subjectinput[f] == Course_Allocation[l][j][k])
            {

               if not dayPrinted
               {
                   write day + "," to outfile
                   dayPrinted = true
                   printedDays[k] = true
               }
                else{
                   write "," to outfile
                }

               if(not timePrinted)
               {
                   write time + "," to outfile
                   timePrinted = true
                   printedTimes[j] = true
               }
                else{
                   write "," to outfile
                }


               write subjectinput[f] to outfile
               write professor_nameinput[f] to outfile
               write branch[branchinput[f]] to outfile
```

```
                    if i == 0:
                        write "LT1" to outfile
                    else if i == 1:
                        write "LT2" to outfile
                    # Continue writing room names for other indices...
                    else:
                        write "CEP-" + (i + 1) to outfile

                    write newline to outfile
                    slotPrinted = true
                }
                if not slotPrinted and dayPrinted
                {
                    write  time to outfile
                }
            }
        }
    }

  close outfile
}
```