

Data Structure Final Project



By: Hansel Faren (2501990350)

Kevin Matthew Tanuwijaya (2502036514)

Nathaniel Davide Darmawan (2502087830)

L2BC / Computer Science

Problems Description

As for a startup company or some people that are planning to start a business, they need some preparation to start their business. One of them is when they have some employees or staff, they will need some program to store the data of the employee or staff that is working for their business. As for the solution, we came up with a program where they can store their employees or staff's information.

Choice of Data Structure

The data structure that we use is vector. Why we used vector is because vector is a one-dimensional data structure and easier access to the memory. Even though it consumes more memory, vector can manage the memory and also can extend dynamically.

Alternative Data Structure

Other data structures that can be used in our program would be arrays or linked lists. Array consumes a small amount of memory because its memory cannot extend dynamically where it's not suited for our program. On the other hand, linked lists are the best when it comes to insertion and deletion elements in the middle but it is not the best when you want to access a known index.

Program Manual

This database program uses a command line interface. A menu will appear when you start the program, and it will prompt you with 5 actions.

```
=====EMPLOYEE DATABASE=====
1. View Database
2. Insert new employee
3. Edit employee data
4. Delete employee data
5. Close

Input an action: 
```

- 1.) Here the program prints every value in the database in an set order and you will be able to view your database.

```
6
7 class Database{
8     //all variables
9     int id;
10    string name;
11    string address;
12    string BirthDate;
13    string PhoneNumber;
14    string position;
15
16    int HoursWorked;
17    int HoursOvertime;
18
19    int RateNormal;
20    int RateOvertime;
21    int salary;
22
23    //making vectors
24    public:
25    vector<int> EmpID;
26
27    vector<string> EmpName;
28    vector<string> EmpAddress;
29    vector<string> EmpBirthDate;
30    vector<string> EmpPhoneNumber;
31    vector<string> EmpPosition;
32
33    vector<int> EmpHoursWorked;
34    vector<int> EmpHoursOvertime;
35
36    vector<int> EmpRateNormal;
37    vector<int> EmpRateOvertime;
38    vector<int> EmpSalary;
```

- 2.) This is the setters of the value

```
40 //all setters
41 void set_id(int x){
42     id=x;
43 }
44
45 void set_name(string x){
46     name=x;
47 }
48
49 void set_address(string x){
50     address=x;
51 }
52
53 void set_BDate(string x){
54     BirthDate=x;
55 }
56
57 void set_number(string x){
58     PhoneNumber=x;
59 }
60
61 void set_position(string x){
62     position=x;
63 }
```

```

64
65     void set_Hours(int x){
66         HoursWorked=x;
67     }
68
69     void set_overtime(int x){
70         HoursOvertime=x;
71     }
72
73     void set_rate(int x){
74         RateNormal=x;
75     }
76
77     void set_rateOver(int x){
78         RateOvertime=x;
79     }
80
81     void set_salary(int x){
82         salary=x;
83     }
84

```

3.) This class to run some function that is featured in our program

```

85
86     //all the getters made will be used to input the info there (exclude hours and salary)
87     //note: we can force them to make an if statement calling all setters
88     void employee_info_store(int id, string name, string address, string BirthDate, string PhoneNumber, string position){
89         EmpID.push_back(id);
90         EmpName.push_back(name);
91         EmpAddress.push_back(address);
92         EmpBirthDate.push_back(BirthDate);
93         EmpPhoneNumber.push_back(PhoneNumber);
94         EmpPosition.push_back(position);
95         EmpSalary.push_back(0);
96     }
97
98     //check their total hours worked
99     void Total_hours(int HRS, int HRSO){
100         EmpHoursWorked.push_back(HRS);
101         EmpHoursOvertime.push_back(HRSO);
102     }
103
104     //calculate their salary, maybe an if statement if we have time
105     void Calculate_Salary(int x){
106         x-=1;
107         int NormalHr=EmpHoursWorked[x];
108         int OvertimeHr=EmpHoursOvertime[x];
109         int total= NormalHr*RateNormal+OvertimeHr*NormalHr;
110         EmpSalary[x]=total;
111     }
112

```

4.) This class is for showing that the database is updated and for removing value

```

114     void updatedMessage(){
115         cout << "=====Database successfully updated!=====";
116     }
117
118     //a way to remove employees by their index (not sorting proof)
119     void remove_employee(int x){
120         x-=1;
121         EmpName.erase(EmpName.begin()-1+x);
122         EmpAddress.erase(EmpAddress.begin()-1+x);
123         EmpBirthDate.erase(EmpBirthDate.begin()-1+x);
124         EmpPhoneNumber.erase(EmpPosition.begin()-1+x);
125         EmpPosition.erase(EmpPosition.begin()-1+x);
126
127         EmpHoursWorked.erase(EmpHoursWorked.begin()-1+x);
128         EmpHoursOvertime.erase(EmpHoursOvertime.begin()-1+x);
129
130         EmpRateNormal.erase(EmpRateNormal.begin()-1+x);
131         EmpRateOvertime.erase(EmpRateOvertime.begin()-1+x);
132         EmpSalary.erase(EmpSalary.begin()-1+x);
133     }
134
135 };

```

5.) Starts the driver file with declaring variable

```
1  /*
2  Driver for the Command Line Interface
3  */
4  #include <iostream>
5  #include <vector>
6  #include <./Database.hpp>
7
8  int main()
9  {
10     int id;
11     string name;
12     string address;
13     string BirthDate;
14     string PhoneNumber;
15     string position;
16
17     int HoursWorked;
18     int HoursOvertime;
19
20     int RateNormal;
21     int RateOvertime;
22     int salary;
23
24     Database EMPLOYEE;
25     int userInput = 0;
26     int userInput2 = 0;
27     int i = -1;
```

6.) While function to make sure the program runs endlessly except choose the terminate function

```
29     while (userInput != 5){
30         cout<<"\n=====EMPLOYEE DATABASE=====";
31         cout<<"\n1. View Database\n2. Insert new employee\n3. Edit employee data\n4. Delete employee data\n5. Close";
32         cout<<"\n\nInput an action: ";
33         cin >> userInput;
34
35         switch (userInput){
36             case 1:
37                 cout << "\n INDEX//ID//NAME//ADDRESS//BIRTH DATE//PHONE NUMBER//POSITION//HOURS WORKED//HOURS WORKED OVERTIME//NORMAL RATE//OVERTIME RATE\n";
38                 for(int x = 0; x < EMPLOYEE.EmpID.size(); x++){
39                     cout<<x << ". // "<<EMPLOYEE.EmpID[x] << "/"<<EMPLOYEE.EmpName[x] << "/"<<EMPLOYEE.EmpAddress[x]<< "/"<<EMPLOYEE.EmpBirthDate[x]<< "/"<< EMPLOYEE.EmpPh
40                 }
41                 break;
42
43             case 2:
44
45
46                 cout << "\nInput ID: ";
47                 cin >> id;
48                 while (id<0){
49                     cout << "Value must be a positive integer: ";
50                     cin >> id;
51                 }
52                 EMPLOYEE.EmpID.push_back(id);
53
54                 cout << "Input Name: ";
55                 cin >> name;
56                 EMPLOYEE.EmpName.push_back(name);
57
58                 cout << "Input Address: ";
59                 cin >> address;
60                 EMPLOYEE.EmpAddress.push_back(address);
61
```

```

61
62     cout << "Input Birth Date: ";
63     cin >> BirthDate;
64     EMPLOYEE.EmpBirthDate.push_back(BirthDate);
65
66     cout << "Input Phone Number: ";
67     cin >> PhoneNumber;
68     EMPLOYEE.EmpPhoneNumber.push_back(PhoneNumber);
69
70     cout << "Input Position: ";
71     cin >> position;
72     EMPLOYEE.EmpPosition.push_back(position);
73
74     cout << "Input Hours Worked: ";
75     cin >> HoursWorked;
76     while (HoursWorked<0){
77         cout << "Value must be a positive integer: ";
78         cin >> HoursWorked;
79     }
80     EMPLOYEE.EmpHoursWorked.push_back(HoursWorked);
81
82     cout << "Input Hours Worked Overtime: ";
83     cin >> HoursOvertime;
84     while (HoursOvertime<0){
85         cout << "Value must be a positive integer: ";
86         cin >> HoursOvertime;
87     }
88     EMPLOYEE.EmpHoursOvertime.push_back(HoursOvertime);
89

```

```

90     cout << "Input Normal Rate: ";
91     cin >> RateNormal;
92     while (RateNormal<0){
93         cout << "Value must be a positive integer: ";
94         cin >> RateNormal;
95     }
96     EMPLOYEE.EmpRateNormal.push_back(RateNormal);
97
98     cout << "Input Overtime Rate: ";
99     cin >> RateOvertime;
100    while (RateOvertime<0){
101        cout << "Value must be a positive integer: ";
102        cin >> RateOvertime;
103    }
104    EMPLOYEE.EmpRateOvertime.push_back(RateOvertime);
105
106    EMPLOYEE.updatedMessage();
107    break;
108

```

```

109    case 3:
110        if(EMPLOYEE.EmpID.size() == 0){
111            cout << "The database is still empty!";
112            break;
113        }
114
115        i = -1;
116        userInput2 = 1;
117        cout << "Enter the employee index of the employee you want to edit: ";
118        cin >> i;
119        if (i<0 || i>EMPLOYEE.EmpID.size()){
120            while(i<0 || i>EMPLOYEE.EmpID.size()){
121                cout << "Index must be between 0 and the number of employees: ";
122                cin >> i;
123            }
124        }
125
126        while (userInput2 != 0){
127            cout << "\n Which element would you like to change?";
128            cout << "\n0. Back\n1. ID\n2.Name\n3. Address\n4. DOB\n5. Phone Number\n6. Position\n7.Hours Worked\n8. Overtime Hours\n9. Normal Rate\n10. Overtime Rate";
129            cout << "\nEnter number: ";
130            cin >> userInput2;
131

```

```

202         case 8:
203             cout << "Enter new hours worked overtime: ";
204             cin >> HoursOvertime;
205             while (HoursOvertime<0){
206                 cout << "Value must be a positive integer: ";
207                 cin >> HoursOvertime;
208             }
209             EMPLOYEE.EmpHoursOvertime[i]=HoursOvertime;
210             EMPLOYEE.updatedMessage();
211
212             break;
213
214         case 9:
215             cout << "Enter new normal rate: ";
216             cin >> RateNormal;
217             while (RateNormal<0){
218                 cout << "Value must be a positive integer: ";
219                 cin >> RateNormal;
220             }
221             EMPLOYEE.EmpRateNormal[i]=RateNormal;
222             EMPLOYEE.updatedMessage();
223
224             break;

```

```

131
132         switch (userInput2){
133             case 0:
134                 //cancel edit
135                 cout << "---Cancelling edit---";
136                 break;
137
138             case 1:
139                 cout << "Enter new ID: ";
140                 cin >> id;
141                 while (id<0){
142                     cout << "Value must be a positive integer: ";
143                     cin >> id;
144                 }
145                 EMPLOYEE.EmpID[i] = id;
146                 EMPLOYEE.updatedMessage();
147
148                 break;
149
150             case 2:
151                 cout << "Enter new name: ";
152                 cin >> name;
153                 EMPLOYEE.EmpName[i] = name;
154                 EMPLOYEE.updatedMessage();
155
156                 break;

```

```

157
158             case 3:
159                 cout << "Enter new address: ";
160                 cin >> address;
161                 EMPLOYEE.EmpAddress[i] = address;
162                 EMPLOYEE.updatedMessage();
163
164                 break;
165
166             case 4:
167                 cout << "Enter new DOB: ";
168                 cin >> BirthDate;
169                 EMPLOYEE.EmpBirthDate[i] = BirthDate;
170                 EMPLOYEE.updatedMessage();
171
172                 break;
173
174             case 5:
175                 cout << "Enter new phone number: ";
176                 cin >> PhoneNumber;
177                 EMPLOYEE.EmpPhoneNumber[i] = PhoneNumber;
178                 EMPLOYEE.updatedMessage();
179
180                 break;
181

```

```

181
182         case 6:
183             cout << "Enter new position: ";
184             cin >> position;
185             EMPLOYEE.EmpPosition[i] = position;
186             EMPLOYEE.updatedMessage();
187
188             break;
189
190         case 7:
191             cout << "Enter new hours worked: ";
192             cin >> HoursWorked;
193             while (HoursWorked<0){
194                 cout << "Value must be a positive integer: ";
195                 cin >> HoursWorked;
196             }
197             EMPLOYEE.EmpHoursWorked[i]=HoursWorked;
198             EMPLOYEE.updatedMessage();
199
200             break;
201

```

```

226         case 10:
227             cout << "Enter new overtime rate: ";
228             cin >> RateOvertime;
229             while (RateOvertime<0){
230                 cout << "Value must be a positive integer: ";
231                 cin >> RateOvertime;
232             }
233             EMPLOYEE.EmpRateOvertime[i]=RateOvertime;
234             EMPLOYEE.updatedMessage();
235
236             break;
237
238     }
239 }
240 break;
241

```

```

242         case 4:
243             if(EMPLOYEE.EmpID.size() == 0){
244                 cout << "The database is still empty!";
245                 break;
246             }
247
248             int idtemp = -1;
249             userInput2 = 0;
250             cout << "Enter the employee index of the employee you want deleted: ";
251             cin >> idtemp;
252
253             int checkID=0;
254             bool Checked=false;
255             int empIndex;
256             //making sure the id input is there
257             for(i=0;i<EMPLOYEE.EmpID.size();i++){
258                 checkID=EMPLOYEE.EmpID[i];
259                 if(idtemp==checkID){
260                     Checked=true;
261                     empIndex=i;
262                     break;
263                 }else{
264                     continue;
265                 }
266             }
267

```

```

268             if (Checked==true){
269
270                 while (userInput2 != (1 || 2)){
271                     cout << "Are you sure you want to delete Index " << i;
272                     cout << "\n1. Yes\n2. No" << endl;
273                     cout << "Choice: ";
274                     cin >> userInput2;
275                 }
276                 switch (userInput2){
277                     case 1:
278                         EMPLOYEE.EmpID.erase(EMPLOYEE.EmpID.begin()+empIndex);
279                         EMPLOYEE.EmpName.erase(EMPLOYEE.EmpName.begin()+empIndex);
280                         EMPLOYEE.EmpAddress.erase(EMPLOYEE.EmpAddress.begin()+empIndex);
281                         EMPLOYEE.EmpBirthDate.erase(EMPLOYEE.EmpBirthDate.begin()+empIndex);
282                         EMPLOYEE.EmpPhoneNumber.erase(EMPLOYEE.EmpPhoneNumber.begin()+empIndex);
283                         EMPLOYEE.EmpPosition.erase(EMPLOYEE.EmpPosition.begin()+empIndex);
284
285                         EMPLOYEE.EmpHoursWorked.erase(EMPLOYEE.EmpHoursWorked.begin()+empIndex);
286                         EMPLOYEE.EmpHoursOvertime.erase(EMPLOYEE.EmpHoursOvertime.begin()+empIndex);
287
288                         EMPLOYEE.EmpRateNormal.erase(EMPLOYEE.EmpRateNormal.begin()+empIndex);
289                         EMPLOYEE.EmpRateOvertime.erase(EMPLOYEE.EmpRateOvertime.begin()+empIndex);
290                         break;
291
292                     case 2:
293                         break;
294

```



```

296         } else if (Checked==false){
297             cout << "employee does not exist"<<endl;
298         }
299
300         break;
301
302
303
304         case 5:
305             //breaks switch case; closes program
306             break;
307     }
308 }
309
310
311 return 0;
312
313 }

```

Results

```

C:\Users\Asus\Documents\L28C\Data Structures\with Ms Nuru\driver.exe
-----EMPLOYEE DATABASE-----
1. View Database
2. Insert new employee
3. Edit employee data
4. Delete employee data
5. Close
Input an action: 2
Input ID: 21
Input Name: hansel
Input Address: binus
Input Birth Date: 12
Input Phone Number: 0812
Input Position: manager
Input Hours Worked: 12
Input Hours Worked Overtime: 2
Input Normal Rate: 10
Input Overtime Rate: 2
-----Database successfully updated!-----
-----EMPLOYEE DATABASE-----
1. View Database
2. Insert new employee
3. Edit employee data
4. Delete employee data
5. Close
Input an action: 1
ID//NAME//ADDRESS//BIRTH DATE//PHONE NUMBER//POSITION//HOURS WORKED//HOURS WORKED OVERTIME//NORMAL RATE//OVERTIME RATE
21//hansel//binus//12//0812//manager//12//2//10//2
-----EMPLOYEE DATABASE-----
1. View Database
2. Insert new employee
3. Edit employee data
4. Delete employee data
5. Close
Input an action: 4
Enter the employee index of the employee you want deleted: 0
Are you sure you want to delete Index 0
1. Yes
2. No
Choice: 1

```

```

C:\Users\Asus\Documents\L28C\Data Structures\with Ms Nuru\driver.exe
Input Normal Rate: 10
Input Overtime Rate: 2
-----Database successfully updated!-----
-----EMPLOYEE DATABASE-----
1. View Database
2. Insert new employee
3. Edit employee data
4. Delete employee data
5. Close
Input an action: 1
ID//NAME//ADDRESS//BIRTH DATE//PHONE NUMBER//POSITION//HOURS WORKED//HOURS WORKED OVERTIME//NORMAL RATE//OVERTIME RATE
21//hansel//binus//12//0812//manager//12//2//10//2
-----EMPLOYEE DATABASE-----
1. View Database
2. Insert new employee
3. Edit employee data
4. Delete employee data
5. Close
Input an action: 4
Enter the employee index of the employee you want deleted: 0
Are you sure you want to delete Index 0
1. Yes
2. No
Choice: 1
-----EMPLOYEE DATABASE-----
1. View Database
2. Insert new employee
3. Edit employee data
4. Delete employee data
5. Close
Input an action: 5
-----
Process exited after 35.32 seconds with return value 0
Press any key to continue . . .

```

Answer for question from the day of presentation

1. Delete by ID

Added to the file

2. If a lot of employees are deleted by half, what happens to the memory ?

The memory of the vector cannot be decreased but the memory space that has been used, will be reused for the future elements so that it can be inserted efficiently