

MUHAMMAD HUZAIFA RAMZAN

BSCS-I-B

Submitted to: Dr. Ashfaq hussain farooqi

Oop-lab Assignment 01

Sunday, February 11, 2024

I created a voting management system covering following aspects

1. NA Candidate Management  
2. PA Candidate Management  
3. Voting list Management

This project contains 1 main (.cpp) file and 2 header (.h) files:

1. Oop Assignment 001-1.cpp
2. i- Voter.h ii- Assembly.h

Main menu

Programs starts with a main menu containing 4 options:

1.NA Candidate Management

2.PA Candidate Management

3.Voting List Management

4.Exit

Each option except 4th one i.e exit―which terminates the program― lead to further menus discussed ahead.

|  |
| --- |
|  |
|  |

NA Candidate Management

This menu offers user following options:

1. Add
2. View
3. Back

|  |
| --- |
|  |
|  |

1.Add

|  |
| --- |
|  |
|  |

2.View

|  |
| --- |
|  |
|  |

3.Back

[Back to main menu.](#Mainmenu) (press ctrl + click to follow link)

PA Candidate Management

This menu offers user following options:

1. Add
2. View
3. Back

|  |
| --- |
|  |
|  |

1.Add

|  |
| --- |
|  |
|  |

2.View

|  |
| --- |
|  |
|  |

3.Back

[Back to main menu.](#Mainmenu) (press ctrl + click to follow link)

Voting List Management

1. Add

|  |
| --- |
|  |
|  |

1. View

|  |
| --- |
|  |
|  |

1. Search

|  |  |
| --- | --- |
|  | |
|  |  |

1. Back

[Back to main menu.](#Mainmenu) (press ctrl + click to follow link)

CODE

|  |
| --- |
| **Main.cpp file** |
| #include <iostream>  #include <fstream>  #include "Assembly.h"  #include "Voter.h"  using namespace std;  void mainMenu();  void NAmenu();  void PAmenu();  void votingMenu();  int main()  {  mainMenu();  }  void mainMenu()  {  while (true) {  cout << "Voting Management System\n";  cout << "1.NA Candidate Management \n2.PA Candidate Management \n3.Voting List Management \n4.Exit\n";  int choice;  cin >> choice;  switch (choice)  {  case 1:  cout << "NA Candidate Management\n";  NAmenu();  break;  case 2:  cout << "PA Candidate Management\n";  PAmenu();  break;  case 3:  cout << "Voting List Management\n";  votingMenu();  break;  case 4:  exit(0);  break;  default:  cout << "Choose valid option from menu\n";  }  }  }  void NAmenu()  {  const int size = 50;  assembly NA[size] = { 0 };    while (true) {  ifstream fin;  fin.open("NA.txt");  try {  if (fin.fail())  {  throw runtime\_error("File Not Found.Enter record to create file\n");  }  }  catch (runtime\_error & e)  {  cout << "Error!!" << e.what();  }  int recordsStored = 0;  if (!fin.fail())  {  while (!fin.eof())  {  fin >> NA[recordsStored].number >> NA[recordsStored].name >> NA[recordsStored].cnic >> NA[recordsStored].party;  recordsStored++;  }  recordsStored -= 1;  }  cout << "1.Add \n2.View \n3.Back\n";  int choice;  cin >> choice;  switch (choice)  {  case 1:  cout << "Calling Add function\n";  add(NA, recordsStored, 1);  break;  case 2:  cout << "Calling View function\n";  view(NA, recordsStored);  break;  case 3:  mainMenu();  break;  default:  cout << "Choose valid option from menu\n";  }  }  }  void PAmenu()  {  const int size = 50;  assembly PA[size] = { 0 };    while (true) {  ifstream fin;  fin.open("PA.txt");  try {  if (fin.fail())  {  throw runtime\_error("File Not Found.Enter record to create file\n");  }  }  catch (runtime\_error & e)  {  cout << "Error!!" << e.what();  }  int recordsStored = 0;  if (!fin.fail())  {  while (!fin.eof())  {  fin >> PA[recordsStored].number >> PA[recordsStored].name >> PA[recordsStored].cnic >> PA[recordsStored].party;  recordsStored++;  }  recordsStored -= 1;  }  cout << "1.Add \n2.View \n3.Back\n";  int choice;  cin >> choice;  switch (choice)  {  case 1:  cout << "Calling Add function\n";  add(PA, recordsStored, 0);  break;  case 2:  cout << "Calling View function\n";  view(PA, recordsStored);  break;  case 3:  mainMenu();  break;  default:  cout << "Choose valid option from menu\n";  }  }  }  void votingMenu()  {  const int size = 50;  voter Voters[size] = { 0 };    while (true) {  ifstream fin;  fin.open("Voters\_Detail.txt");  try {  if (fin.fail())  {  throw runtime\_error("File Not Found.Enter record to create file\n");  }  }  catch (runtime\_error & e)  {  cout << "Error!!" << e.what();  }  int recordsStored = 0;  if (!fin.fail())  {  while (!fin.eof())  {  fin >> Voters[recordsStored].cnic >> Voters[recordsStored].area >> Voters[recordsStored].NAnumber >> Voters[recordsStored].PAnumber >> Voters[recordsStored].blockCode;  recordsStored++;  }  recordsStored -= 1;  }  cout << "1.Add \n2.View \n3.Search \n4.Back\n";  int choice;  cin >> choice;  switch (choice)  {  case 1:  cout << "Calling Add function\n";  add(Voters, recordsStored);  break;  case 2:  cout << "Calling View function\n";  view(Voters, recordsStored);  break;  case 3:  cout << "Calling Search function\n";  uint64\_t required;  cout << "Enter CNIC to search ";  cin >> required;  bool flag;  flag= search(Voters, recordsStored, required);  if (flag)  cout << "voter exists\n";  else  cout << "Record not found\n";  break;  case 4:  mainMenu();  break;  default:  cout << "Choose valid option from menu\n";  }  }  } |

|  |
| --- |
| **Assembly.h file** |
| #pragma once  using namespace std;  struct assembly {  int number;  string name, party, cnic;  };  void add(assembly p[], int index ,bool signal) {  cout << "Enter HALQA NUMBER\t: ";  cin >> p[index].number;  cout << "Enter Candidate's CNIC\t: ";  cin >> p[index].cnic;  cout << "Enter Candidate's Name\t: ";  cin >> p[index].name;  cout << "Enter Candidate's Party\t: ";  cin >> p[index].party;  ofstream fout;  if (signal)  {  fout.open("NA.txt");  }  else  {  fout.open("PA.txt");  }  for (int i = 0; i < index+1; i++)  {  fout << p[i].number << "\t" << p[i].name << "\t"<< p[i].cnic << "\t" << p[i].party << "\n";  }  fout.close();  }  void view(const assembly p[], int index) {  cout << "HALQA\tCNIC\t\tName\tParty\n";  for (int i = 0; i < index; i++)  {  cout << p[i].number << "\t" << p[i].cnic << "\t" << p[i].name << "\t" << p[i].party<<"\n";  }  } |

|  |
| --- |
| Voter.h file |
| #pragma once  using namespace std;  struct voter {  uint64\_t cnic;  string area;  int NAnumber, PAnumber, blockCode;  };  void add(voter p[], int index)  {  cout << "Enter Voter's CNIC\t: ";  cin >> p[index].cnic;  cout << "Enter Area\t\t: ";  cin >> p[index].area;  cout << "Enter NA Number\t\t: ";  cin >> p[index].NAnumber;  cout << "Enter PA Number\t\t: ";  cin >> p[index].PAnumber;  cout << "Enter Block Code\t: ";  cin >> p[index].blockCode;  ofstream fout;  fout.open("Voters\_Detail.txt");  for (int i = 0; i < index + 1; i++)  {  fout << p[i].cnic << "\t" << p[i].area << "\t" << p[i].NAnumber << "\t" << p[i].PAnumber << "\t" << p[i].blockCode << "\n";  }  fout.close();  }  void view(voter p[], int index)  {  cout << "CNIC\t\tArea\tNA#\tPA#\tBlockCode\n";  for (int i = 0; i < index; i++)  {  cout << p[i].cnic << "\t" << p[i].area << "\t" << p[i].NAnumber << "\t" << p[i].PAnumber << "\t" << p[i].blockCode << "\n";  }  }  bool search(voter p[], int index, uint64\_t required)  {  for (int i = 0; i < index; i++)  {  if (p[i].cnic==required)  {  return 1;  }  }  return 0;  } |