TRƯỜNG ĐẠI HỌC CÔNG NGHỆ SÀI GÒN

**KHOA CÔNG NGHỆ THÔNG TIN**

🙤🙦🙤🙦

**BÁO CÁO**

**ĐỒ ÁN TIN HỌC**

Người hướng dẫn: ThS. Nguyễn Thị Ngân Hà

Sinh viên thực hiện:

Họ tên: Nguyễn Ngọc Thiện

MSSV: DH52107203

Lớp: D21\_TH01

TP. Hồ Chí Minh – Năm 2023

Mục Lục

Phần 1: Phần Cơ Sở (4.5đ) …………………………………………………………………………………………………………3

Cài Đặt Cấu Trúc Cây BST

Phần 2: Phần Vận Dụng (4.5đ) …………………………………………………………………………………………………..7

Quản Lý Thông Tin Nhân Viên

Phần 3: Phần Mở Rộng (1đ) ..…………………………………………………………………………………………………22

Thuật Toán Mã Hóa A5/1

**Phần I: Phần Cơ Sở (4.5 điểm):**

Câu 5: Cài đặt cấu trúc cây BST

Ý tưởng làm bài: sử dụng ngôn ngữ C++ và Lập trình Console

Nội dung: Cây BST – Cây tìm kiếm nhị phân (Binary Search Tree) là một cấu trúc dữ liệu rất thuận lợi cho bài toán tìm kiếm

Định nghĩa:

* Cây tìm kiếm ứng với n khóa k1, k2, k3, … là cây nhị phân mà mỗi Nút đều được gán một khóa sao cho với mỗi Nút k:
* Mọi khóa trên cây con trái đều nhỏ hơn khóa trên Nút k
* Mọi khóa trên cây con phải đều lớn hơn khóa trên Nút k
* Cây tìm kiếm nhị ohân là một cấu trúc dữ liệu cơ bản được sử dụng để xây dựng các cấu trúc dữ liệu trừu tượng hơn như các tập hợp, đa tập hợp, các dãy kết hợp
* Nếu một BST có chứa các giá trị giống nhau thì nó biểu diễn môt đa tập hợp. Cây loại này sử dụng các bất đẳng thức không nghiêm ngặt. một Nút trong cây con trái có khóa nhỏ hơn khóa của Nút cha, mọi Nút trên cây cây con phải có Nút lớn hơn hoặc bằng khóa của Nút cha

Code Build: Chủ yếu nhập các số tự nhiên không trùng nhau vào cây BST

Thành phần:

|  |  |
| --- | --- |
| **Cấu trúc của Nút:** | **Khởi tạo Cây BST:** |
| struct Node  {  int data;  Node\* right;  Node\* left;  }; | void createTree\_BST(Tree\_BST& t)  {  t = NULL;  } |

|  |  |
| --- | --- |
| **Khởi tạo Nút:** gồm các số nguyên tố không trùng nhau | **Thêm Nút vào Cây BST:** |
| void create\_Node(Tree\_BST& t, int x)  {  if (t == NULL)  {  Node\* p = new Node();  p->data = x;  p->left = NULL;  p->right = NULL;  t = p;  }  else  {  if (x < t->data)  {  create\_Node(t->left, x);  }  else if (x > t->data)  {  create\_Node(t->right, x);  }  else  cout << "\nTon tai Node";  }  } | void Insert\_Node\_Tree(Tree\_BST& t)  {  int x;  cout << "\nNhan -1 de dung them phan tu vao Cay BST";  while (true)  {  cout << "\nThem phan tu: ";  cin >> x;  if (x == -1)  break;  create\_Node(t, x);  }  } |

**Xuất Cây BST ra màng hình Console**

|  |  |  |
| --- | --- | --- |
| PreOrder | InOrder | PostOrder |
| void PreOrder(Tree\_BST t)  {  if (t != NULL)  {  cout << t->data << " ";  PreOrder(t->left);  PreOrder(t->right);  }  } | void InOrder(Tree\_BST t)  {  if (t != NULL)  {  InOrder(t->left);  cout << t->data << " ";  InOrder(t->right);  }  } | void PostOrder(Tree\_BST t)  {  if (t != NULL) {  PostOrder(t->left);  PostOrder(t->right);  cout << t->data << " ";  }  } |

**check\_PrimeNumber:** Kiểm tra số nguyên tố có trong cây BST

|  |
| --- |
| bool check\_PrimeNumber(int x)  {  if (x < 2)  {  return false;  }  if (x == 2)  return true;  for (int i = 2; i < x; i++)  {  if (x % i == 0)  return false;  }  return true;  } |

**Count\_PrimeNumber:** Đếm số nguyên tố có trong cây

|  |
| --- |
| void Count\_PrimeNumber(Tree\_BST t, int& count\_primeNumber)  {  if (t != NULL)  {  if (check\_PrimeNumber(t->data))  {  count\_primeNumber++;  }  Count\_PrimeNumber(t->left, count\_primeNumber);  Count\_PrimeNumber(t->right, count\_primeNumber);  }  } |

**Output\_PreimeNumber:** Xuất các số nguyên tố có trong cây

|  |
| --- |
| void Output\_PrimeNumber(Tree\_BST t)  {  if (t != NULL)  {  if (check\_PrimeNumber(t->data))  cout << t->data << " ";  Output\_PrimeNumber(t->left);  Output\_PrimeNumber(t->right);  }  } |

**Search\_Node:** Tìm kiếm Nút có trong cây

|  |
| --- |
| Node\* Search\_Node(Tree\_BST &t, int x)  {  if (t == NULL || t->data == x)  return t;  if (x < t->data)  return Search\_Node(t->left, x);  else if (x > t->data)  return Search\_Node(t->right, x);  } |

**Output\_Node\_La**: xuất Nút lá

|  |
| --- |
| void Output\_Node\_La(Tree\_BST t)  {  if (t != NULL)  {  if (t->left == NULL && t->right == NULL)  {  cout << t->data << " ";  }  Output\_Node\_La(t->left);  Output\_Node\_La(t->right);  }  } |

**Output\_Node\_1**: Xuất Nút có 1 con

|  |
| --- |
| void Output\_Node\_1(Tree\_BST t)  {  if (t != NULL)  {  if ((t->left != NULL && t->right == NULL) || (t->left == NULL && t->right != NULL))  {  cout << t->data << " ";  }  Output\_Node\_1(t->left);  Output\_Node\_1(t->right);  }  } |

**Output\_Node\_2:** Xuất Nút có 2 con

|  |
| --- |
| void Output\_Node\_2(Tree\_BST t)  {  if (t != NULL)  {  if (t->left != NULL && t->right != NULL)  {  cout << t->data << " ";  }  Output\_Node\_2(t->left);  Output\_Node\_2(t->right);  }  } |

|  |  |
| --- | --- |
| **Node\_Max:** Tìm Nút có giá trị lớn nhất | **Node\_Min:** Tìm Nút có giá trị nhỏ nhất |
| int Node\_Max(Tree\_BST t)  {  if (t->left == NULL && t->right == NULL)  {  return t->data;  }  return Node\_Max(t->right);  } | int Node\_Min(Tree\_BST t)  {  if (t->left == NULL && t->right == NULL)  {  return t->data;  }  return Node\_Max(t->left);  } |

**LeftMostChild:** Tìm Nút thế mạng ở cây con trái

|  |
| --- |
| void LeftMostChild(Tree\_BST&X,Tree\_BST&Y)  {  while (Y->left!=NULL)  {  Y = Y->left;  }  X->data = Y->data;  X = Y;  Y = Y->right;  } |

|  |  |
| --- | --- |
| **Delete\_Node:** Xóa Nút | **Remove\_BST:** Hủy Cây |
| void Delete\_Node(Tree\_BST& t, int node)  {  if (t == NULL)  return;  if (t->data > node)  Delete\_Node(t->left, node);  else if (t->data < node)  Delete\_Node(t->right, node);  else  {  Node\* X = t;  if (t->left == NULL)  t = t->right;  else if (t->right == NULL)  t = t->left;  else  {    LeftMostChild(X, t->right);  }  delete X;  }  } | void Remove\_BST(Tree\_BST& t)  {  if (t == NULL)  return;  Remove\_BST(t->left);  Remove\_BST(t->right);  delete t;  t = NULL;  } |

**Phần II: Phần Vận Dụng (4.5 điểm):**

Bài 3: Một công ty cần quản lí thông tin của nhân viên như sau:

+ Tên nhân viên, ngày sinh, quê quán, mức lương cơ bản, hệ số lương.

+ Lương của nhân viên được tính dựa trên công thức: lương cơ bản \* hệ số lương

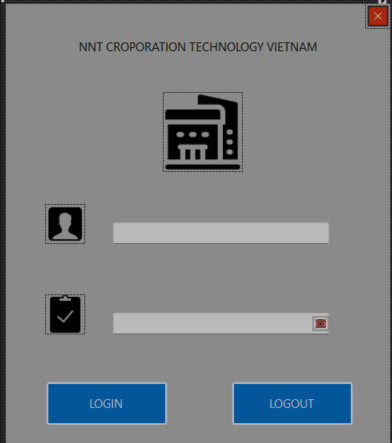
+ Sử dụng cấu trúc dữ liệu phù hợp để lưu trữ thông tin nhân viên thực hiện các yêu cầu sau:

1. Thêm/ xóa/ sửa thông tin của nhân viên
2. Cho biết nhân viên nào có lương cao nhất, nhân viên nào có lương thấp nhất
3. Sắp xếp danh sách nhân viên theo thứ tự tăng/ giảm dần dựa trên mức lương

Ý tưởng làm bài: Dùng ngôn ngữ C# và Lập trình Window Forms App

**Desgin Demo:**

**From Login:**



Thành Phần:

**txtUsername**: Nhập tài khoản vào ( Username : “admin”);

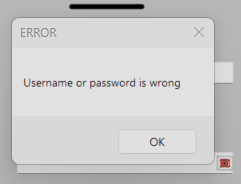
**txtPasswork**: Nhập mật khẩu; hiện trên màn hình là dấu \* (Password: “123”)

**Button LOGIN:**

|  |
| --- |
| private void bntLogIn\_Click(object sender, EventArgs e)  {  string user = txtUser.Text;  string pass = txtPass.Text;  if (user != User || pass != Password)  {  MessageBox.Show("Username or password is wrong", "ERROR");  return;  }  C\_common.save\_user = txtUser.Text;  C\_common.save\_pass = txtPass.Text;  frmInfoEmployee Manager = new frmInfoEmployee();  Manager.Show();  this.Hide();  } |

+ Nhập đúng Username và Password thì From Information Employee xuất hiện và From Login được ẩn đi

+ Nhập sai Username hoặc Password thì bảng thông báo lỗi sẽ hiện ra



**Button LOGOUT:** Thoát chương trình hiện hành

|  |
| --- |
| private void bntLogOut\_Click(object sender, EventArgs e)  {  var result = MessageBox.Show("LOGED OUT", "Notification", MessageBoxButtons.OKCancel);  if (result == DialogResult.Cancel)  {  return;  }  Application.Exit();  } |

**From Information Employee:**

****

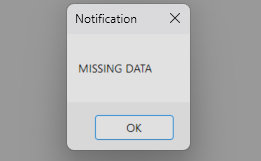
Thành Phần:

**Button INSERT:**

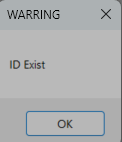
|  |
| --- |
| private void bntInsert\_Click(object sender, EventArgs e)//Thêm nhân viên vào danh sách  {  if (txtID.Text == "" || txtName.Text == "" || cmbPosition.Text == "" || txtAddress.Text == "" || txtPhoneNumber.Text == "" || txtEmail.Text == "")  {  MessageBox.Show("MISSING DATA", "Notification", MessageBoxButtons.OK);  return;  }  if (check\_ID(txtID.Text) != null)  {  MessageBox.Show("ID Exit", "WARRING");  clear\_Text();  return;  }  C\_InfoEmployee emp = new C\_InfoEmployee()  {  ID = txtID.Text,  Name\_Employee = txtName.Text,  Gender = Gender\_show(),  DateOfBirth = dtpDateOfBirth.Value,  Position = cmbPosition.Text,  Address = txtAddress.Text,  PhoneNumber = txtPhoneNumber.Text,  Email = txtEmail.Text,  DateStartWork = dtpDateStartWork.Value,  Achievements = txtAchievements.Text,  Salary = "",  Note = txtNote.Text  };  C\_common.clss\_Infor.Add(emp);  show\_list\_infor\_Employee();  clear\_Text();  Check\_Save\_Click = false;// chưa lưu  Empty\_Data = false;//danh sach khong rong  } |

+ Dữ liệu được nhập từ các text box được đưa vào datagirdview

+ Nhập thiếu dữ liệu ( VD: Thiếu “Phone Number” ) bảng thông báo lỗi hiện ra



+ Nếu nhập trùng “ID”: hiện thông báo



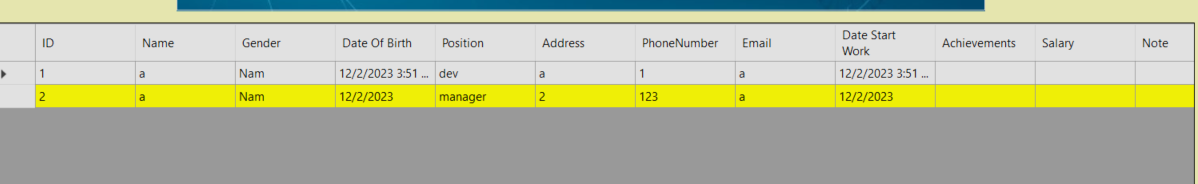
* Hàm dùng để kiểm tra “ID” (Nhân viên) có trong danh sách hay không

|  |
| --- |
| public C\_InfoEmployee check\_ID(string id)// kiểm tra ID  {  foreach (C\_InfoEmployee Employee in C\_common.clss\_Infor)  {  if (Employee.ID == id)  return Employee;  }  return null;  } |

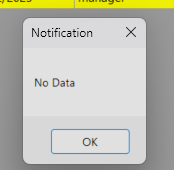
**Button SEARCH:** Tìm kiếm theo ID

|  |
| --- |
| private void bntSearch\_Click(object sender, EventArgs e)//Tìm ID  {  C\_InfoEmployee search\_ID = check\_ID(txtSearchID.Text);  if (search\_ID == null)  {  MessageBox.Show("No Data", "Notification");  txtSearchID.Clear();  return;  }  Refersh\_Color();  foreach (DataGridViewRow row in dgvInformation.Rows)  {  C\_InfoEmployee employee = row.DataBoundItem as C\_InfoEmployee;  if (employee == search\_ID)  {  // Tô màu dòng chứa đối tượng search\_ID  row.DefaultCellStyle.BackColor = Color.Yellow;  dgvInformation.FirstDisplayedScrollingRowIndex = row.Index;  dgvInformation.Refresh();  break;  }  }  txtSearchID.Clear();  } |

+ Tìm thấy: Dòng có ID đó sẽ được tô màu trong datagirdview



+ Không tìm thấy: Xuất bảng thông báo



**Button RELOAD:** Có tác dụng làm dòng được tô màu trở về ban đầu

|  |
| --- |
| private void Refersh\_Color()  {  foreach (DataGridViewRow row in dgvInformation.Rows)  {  row.DefaultCellStyle.BackColor = Color.White;  }  }  private void bntReload\_Click(object sender, EventArgs e)//Trả về danh sách nhân viên  {  Refersh\_Color();  dgvInformation.ClearSelection();  } |

* Sự kiện Cell\_Click trong bảng Datagirdview

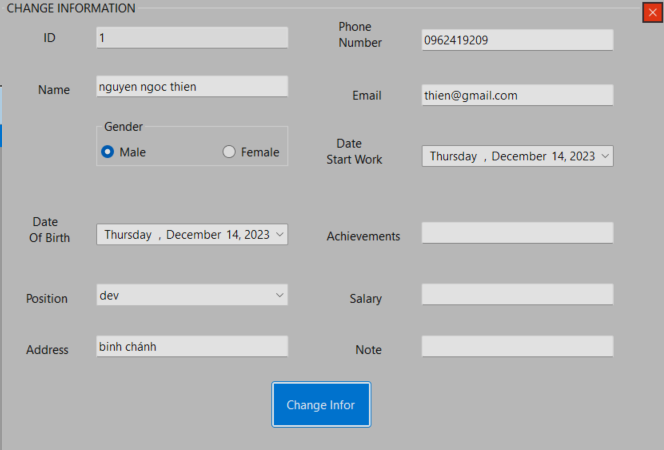
|  |
| --- |
| private void dgvInforEmployee\_Cell\_Click(object sender, DataGridViewCellEventArgs e)  {  if (e.RowIndex != -1)  {  rowIndex = dgvInformation.SelectedRows[0].Index;  choseRow = true;  changeInformation = new frmChangeInformation(rowIndex);  C\_InfoEmployee inf = C\_common.clss\_Infor[e.RowIndex];  changeInformation.txtID\_Change.Text = inf.ID;  changeInformation.txtName\_Change.Text = inf.Name\_Employee;  if (inf.Gender == "Nữ")  {  changeInformation.rbtFemale\_Change.Checked = true;  }  changeInformation.txtAddress\_Change.Text = inf.Address;  changeInformation.dtpDateOfBirth\_Change.Value = inf.DateOfBirth;  changeInformation.cmbPosition\_Change.Text = inf.Position;  changeInformation.txtPhoneNumber\_Change.Text = inf.PhoneNumber;  changeInformation.txtEmail\_Change.Text = inf.Email;  changeInformation.dtpDateStartWork\_Change.Value = inf.DateStartWork;  changeInformation.txtAchivements\_Change.Text = inf.Achievements;  changeInformation.txtSalary\_Change.Text = inf.Salary;  changeInformation.txtNote\_Change.Text = inf.Note;  }  } |

**Button CHANGE INF:**

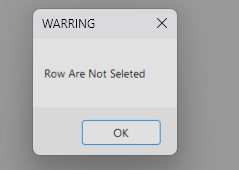
|  |
| --- |
| private void bntfix\_Click(object sender, EventArgs e)  {  if (choseRow)  {  changeInformation.ShowDialog();  show\_list\_infor\_Employee();  DecorColor(rowIndex);  choseRow = false;  Empty\_Data = false;  Check\_Save\_Click = false;  }  else  MessageBox.Show("Row Are Not Seleted", "WARRING");  } |

+ Chọn 1 dòng muốn sửa dữ liệu, bấm vào button này, Form Change Information Employee hiện ra cùng với thông tin của dòng đó. (Nhờ sự kiện Cell\_Click)

**From Change Information Employee:**



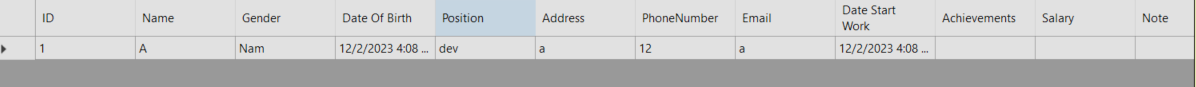
+ Chưa chọn dòng, bấm button này, hiện bảng thông báo



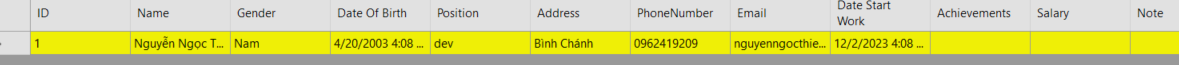
+ Sau khi thông tin đã được thay đổi, nhấn button Change Infor, thì bảng “Datagirdview” được cập nhật lại dữ liệu mới, đồng thời đóng frmChangeInforEmployee và dòng chứa thông tin đó được tô màu

|  |
| --- |
| private void bntChangeInf\_Click(object sender, EventArgs e)  {  C\_common.clss\_Infor[SelectedRowIndex].Name\_Employee = txtName\_Change.Text;  C\_common.clss\_Infor[SelectedRowIndex].DateOfBirth = dtpDateOfBirth\_Change.Value;  C\_common.clss\_Infor[SelectedRowIndex].Position = cmbPosition\_Change.Text;  C\_common.clss\_Infor[SelectedRowIndex].Address = txtAddress\_Change.Text;  C\_common.clss\_Infor[SelectedRowIndex].PhoneNumber = txtPhoneNumber\_Change.Text;  C\_common.clss\_Infor[SelectedRowIndex].Email = txtEmail\_Change.Text;  C\_common.clss\_Infor[SelectedRowIndex].DateStartWork = dtpDateStartWork\_Change.Value;  C\_common.clss\_Infor[SelectedRowIndex].Achievements = txtAchivements\_Change.Text;  C\_common.clss\_Infor[SelectedRowIndex].Salary = txtSalary\_Change.Text;  C\_common.clss\_Infor[SelectedRowIndex].Note = txtNote\_Change.Text;  if (rbtMale\_Change.Checked == true)  {  C\_common.clss\_Infor[SelectedRowIndex].Gender = "Nam";  }  else  C\_common.clss\_Infor[SelectedRowIndex].Gender = "Nữ";  MessageBox.Show("Data Have Been Changed", "Notification");  clear\_Change\_Text();  this.Close();  } |

Dữ liệu ban đầu:



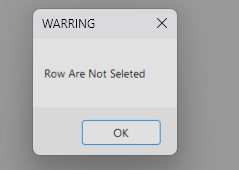
Dữ liệu được thay đổi:



**Button DELETE:** Xóa dòng được chọn

|  |
| --- |
| private void bntDelete\_Click(object sender, EventArgs e)  {  if (dgvInformation.SelectedRows.Count > 0)  {  int index = dgvInformation.SelectedRows[0].Index;  C\_common.clss\_Infor.RemoveAt(index);  choseRow = false;  show\_list\_infor\_Employee();  }  else  MessageBox.Show("Row Are Not Seleted");  } |

+ Chưa chọn dòng, xuất bảng thông báo



* Hàm dùng để lưu dữ liệu vào tệp “txt”.

|  |
| --- |
| private void SaveFile(string filename)  {  try  {  StringBuilder sb = new StringBuilder();  foreach (DataGridViewRow row in dgvInformation.Rows)  {  foreach (DataGridViewCell cell in row.Cells)  {  sb.Append(cell.Value?.ToString() ?? "");  sb.Append("\t\t\t");  }  sb.AppendLine();  }  StreamWriter write = new StreamWriter(filename);  write.WriteLine(sb.ToString());  write.Close();  }  catch (Exception ex)  {  throw ex;  }  } |

**Button Save FILE:** Lưu dữ liệu có trong bảng “datagirdview” vào tệp “txt”.

|  |
| --- |
| private void bntSaveFile\_Click(object sender, EventArgs e)  {  if (Empty\_Data == true)// danh sach rong  {  return;  }  SaveFileDialog dlg = new SaveFileDialog();  dlg.Filter = "Text file (\*.txt)|\*.txt";  if (dlg.ShowDialog() == DialogResult.OK)  {  string filename = dlg.FileName;  SaveFile(filename);  MessageBox.Show("Data is Saved");  }  Check\_Save\_Click = true;// dữ liệu đã lưu  } |

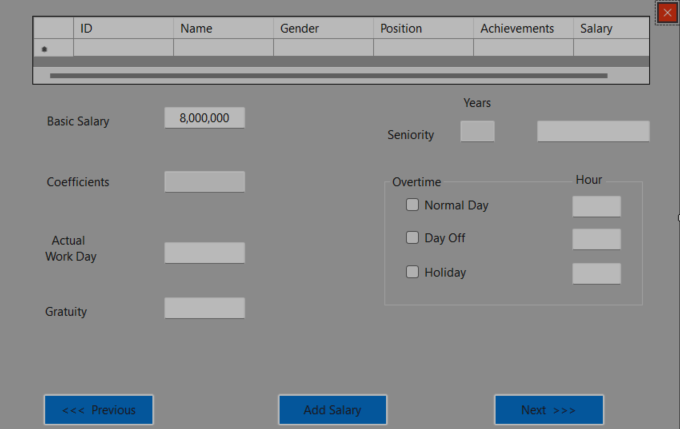
**Button OPEN FILE:** Mở dữ liệu có trong tệp “txt” lên bảng “datagirdview”

|  |
| --- |
| private void bntOpenFile\_Click(object sender, EventArgs e)  {  try  {  OpenFileDialog openfile = new OpenFileDialog();  //Thiết lập hộp thoại mở tệp  openfile.Title = "Chọn tệp tin";  openfile.Filter = "Tệp tin văn bản|\*.txt|Tất cả các tệp tin|\*.\*";  openfile.Multiselect = false;//chỉ được chọn 1 tệp duy nhất  if (openfile.ShowDialog() == DialogResult.OK)  {  string filename = openfile.FileName;  StreamReader reader = new StreamReader(filename);  string line;  while ((line = reader.ReadLine()) != null)  {  string[] value = line.Split("\t\t\t");  //data dc them vao class  if (value.Length == 13)  {  C\_InfoEmployee dataItem = new C\_InfoEmployee()  {  ID = value[0],  Name\_Employee = value[1],  Gender = value[2],  DateOfBirth = DateTime.Parse(value[3]),  Position = value[4],  Address = value[5],  PhoneNumber = value[6],  Email = value[7],  DateStartWork = DateTime.Parse(value[8]),  Achievements = value[9],  Salary = value[10],  Note = value[11],  };  C\_common.clss\_Infor.Add(dataItem);  }  }  reader.Close();  }  show\_list\_infor\_Employee();  Empty\_Data = false;  }  catch (Exception ex)  {  throw ex;  }  } |

**Button CALCULATION OF WAGES:** hiện Form Salary, dùng để tính lương cho nhân viên

|  |
| --- |
| private void CalculationOfWages\_Click(object sender, EventArgs e)  {  if (Empty\_Data == true)  {  return;  }  frmSalary frmsalary = new frmSalary();  C\_Salary.show\_Datagirview\_Salary(0);  frmsalary.dgvSalary.DataSource = C\_Salary.C\_salary.ToList();  frmsalary.txtCoefficients.Text = C\_Salary.Coefficients(0).ToString();  frmsalary.txtYearsSeniority.Text = C\_Salary.YearsSeniority(0).ToString();  frmsalary.ShowDialog();  show\_list\_infor\_Employee();  } |

**Form Salary:**

****

Thành phần:

**Basic Salary:** Lương cơ bản, cụ thể được gán với số tiền là 8,000,000 vnđ

**Coeffiicients:** Hệ số lương - ứng với từng vị trí sẽ có hệ số lương khác nhau

+ Devops: 2.2

+ Manager: 3.1

+ Dev : 1.0

|  |
| --- |
| public static float Coefficients(int employee)  {  float coefficients = 1.0f;  string positon = C\_common.clss\_Infor[employee].Position;  if (positon == "Devops")  return coefficients = 2.2f;  else if (positon == "manager")  return coefficients = 3.1f;  return coefficients;  } |

**Seniority:** Thâm niên - Được tính từ ngày bắt đầu vào làm đến ngày hiện tại (đủ 360 ngày = 1 năm)

+ Nếu số năm thâm niên > =1 thì được phép nhập số tiền thêm cho nhân viên này

|  |
| --- |
| public static int YearsSeniority(int nhanvien)  {  DateTime start = C\_common.clss\_Infor[nhanvien].DateStartWork;  DateTime curren = DateTime.Now;  int seniority = 0;  TimeSpan dura = curren - start;  return seniority = dura.Days / 360;  } |

**Over time:** tăng ca – được tính theo giờ (1h = 25,000 vnđ)

+ Normal day - tăng ca ngày thường: số giờ \* 150/100 \* 25,000 nvđ

+ Day off - tăng ca ngày nghỉ: số giờ \* 200/100 \* 25,000 vnđ

+ Holiday - tăng ca ngày lễ: số giờ \* 300/100 \* 25,000 vnđ

|  |
| --- |
| private float MoneyOverTime()  {  float normalday, dayoff, holiday;  float hourlyWage = 25;  if (txtNormalDay.Text == "")  normalday = 0;  else  normalday = float.Parse(txtNormalDay.Text) \* 150 / 100 \* hourlyWage;  if (txtDayOff.Text == "")  dayoff = 0;  else  dayoff = float.Parse(txtDayOff.Text) \* 200 / 100 \* hourlyWage;  if (txtHoliday.Text == "")  holiday = 0;  else  holiday = float.Parse(txtHoliday.Text) \* 300 / 100 \* hourlyWage;  return normalday + dayoff + holiday;  } |

**Actual Word Day**: số ngày đi làm thực tế

**Gratuity**: tiền thưởng nếu nhân viên có thành tích trong tháng

* Lương = Lương cơ bản \* hệ số lương / 24 \* số ngày đi làm thưc tế + tiền thâm niên + tiền tăng ca + tiền thưởng.

Với 24 là số ngày bắt buộc đi làm tính theo tháng.

|  |
| --- |
| public static string SaLary(int workday,float moneyBacsicSalary, int employee, float gratuity, float moneySeniority, float moneyovertime)  {  float salary = 0;  float coefficients = Coefficients(employee);  salary = coefficients \* moneyBacsicSalary / 24 \* workday + gratuity + moneySeniority+ moneyovertime;  return salary.ToString();  } |

**Button Add Salary**: tiền lương của nhân viên được tính và thêm vào cho nhân viên đó

|  |
| --- |
| private void btnAdd\_Salary\_Click(object sender, EventArgs e)  {  if (txtWorkDay.Text == "")  {  MessageBox.Show("Forget actual work days");  return;  }  if (int.Parse(txtWorkDay.Text) > 24 || int.Parse(txtWorkDay.Text) < 1)  {  MessageBox.Show("Erron: Work Day", "WARRING");  return;  }  int workday = int.Parse(txtWorkDay.Text);  float moneygratuity = MoneyGratuity();  float moneySeniority = MoneySeniority();  float moneyOverTime = MoneyOverTime();  float MoneyBasicSalary = float.Parse(txtBasicSalary.Text);  C\_Salary.C\_salary[0].Salary = C\_Salary.SaLary(workday, MoneyBasicSalary, employee, moneygratuity, moneySeniority, moneyOverTime);  C\_common.clss\_Infor[employee].Salary = C\_Salary.C\_salary[0].Salary;  show\_list\_Salary();  } |

**Button Next:** chuyển đến nhân viên tiếp theo

|  |
| --- |
| private void btnNext\_Click(object sender, EventArgs e)  {  employee++;//nhân viên tăng 1  ItemClear();  if (C\_common.clss\_Infor.Count == employee)//danh sách nhân viên hết  {  var resul = MessageBox.Show("Done");  if (resul == DialogResult.OK)  {  this.Close();  return;  }  }  C\_Salary.show\_Datagirview\_Salary(employee);  show\_list\_Salary();  ItemAdd();  } |

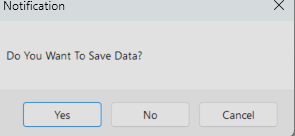
Sau khi tính lương cho tất cả nhân viên, Form Salary đóng, trở về lại Form Information Employee.

**Button Previous:** chuyển đến nhân viên trước đó

|  |
| --- |
| private void btnPrevious\_Click(object sender, EventArgs e)  {  employee--;// nhân viên giảm 1  ItemClear();  if(employee <=0)  {  employee = 0;  C\_Salary.show\_Datagirview\_Salary(0);  }  show\_list\_Salary();  ItemAdd();  } |

**Button EXIT:** Thoát khỏi frmEmployee, trở lại frmLogin

|  |
| --- |
| private void bntExit\_Click(object sender, EventArgs e)//Thoát khỏi frmInfoEmployee, trở về frmLogin  {  if (Check\_Save\_Click == false && Empty\_Data == false)  {  var a = MessageBox.Show("Do You Want To Save Data?", "Notification", MessageBoxButtons.YesNoCancel);  if (a == DialogResult.Cancel)  {  return;  }  if (a == DialogResult.Yes)  {  bntSaveFile.PerformClick();  }  show\_frmLogin();  }  else  {  show\_frmLogin();  }  } |

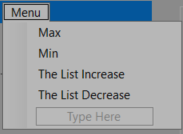
* Nếu dữ liệu đã được lưu hoặc danh sách rỗng
* Thoát frmEmployee, trở về frmLogin
* Nếu dữ liệu chưa được lưu: Hiện bảng thông báo
* 

+ Nhấn “Yes”: lưu dữ liệu và thoát frmEmployee, trở về frmLogin

+ Nhấn “No”: không lưu dữ liệu, thoát frmEmployee, trở về frmLogin

+ Nhấn “Cancel”: thoát bảng thông báo.

Menu:



**+ Max:** tìm nhân viên có số lương cao nhất

|  |
| --- |
| private void maxToolStripMenuItem\_Click\_1(object sender, EventArgs e)  {  if (C\_common.clss\_Infor[0].Salary == "")  return;  int max = int.Parse(C\_common.clss\_Infor[0].Salary);  for (int i = 1; i < C\_common.clss\_Infor.Count; i++)  {  if (max < int.Parse(C\_common.clss\_Infor[i].Salary))  {  max = int.Parse(C\_common.clss\_Infor[i].Salary);  }  }  Refersh\_Color();  print\_Value(max);  } |

**+ Min:** tìm nhân viên có số lương thấp nhất

|  |
| --- |
| private void minToolStripMenuItem\_Click(object sender, EventArgs e)  {  if (C\_common.clss\_Infor[0].Salary == "")  return;  int min = int.Parse(C\_common.clss\_Infor[0].Salary);  for (int i = 1; i < C\_common.clss\_Infor.Count; i++)  {  if (min > int.Parse(C\_common.clss\_Infor[i].Salary))  {  min = int.Parse(C\_common.clss\_Infor[i].Salary);  }  }  Refersh\_Color();  print\_Value(min);  } |

**+ The List Increase:** sắp xếp danh sách theo mức lương từ cao xuống thấp

|  |
| --- |
| private void theListIncreasingToolStripMenuItem\_Click(object sender, EventArgs e)  {  if (C\_common.clss\_Infor[0].Salary == " ")  return;  C\_common.clss\_Infor.Sort((emp1, emp2) => emp2.Salary.CompareTo(emp1.Salary));  dgvInformation.Refresh();  show\_list\_infor\_Employee();  } |

**+ The List Decrease**: sắp xếp danh sách theo mức lương từ thấp đến cao

|  |
| --- |
| private void theListToolStripMenuItem\_Click(object sender, EventArgs e)  {  if (C\_common.clss\_Infor[0].Salary == " ")  return;  C\_common.clss\_Infor.Sort((emp1, emp2) => emp1.Salary.CompareTo(emp2.Salary));  dgvInformation.Refresh();  show\_list\_infor\_Employee();  } |

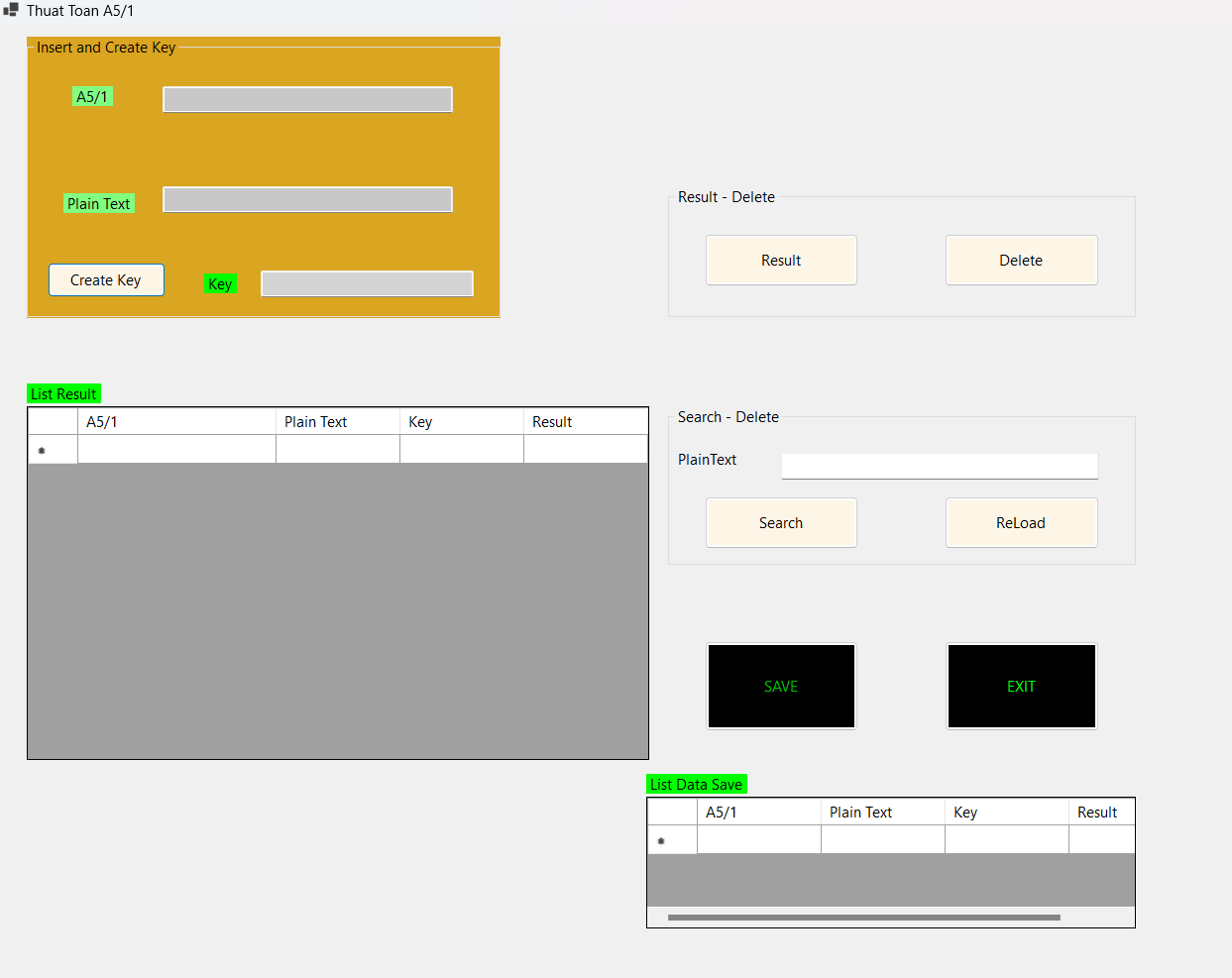
**Phần III: Phần Mở Rộng (1.0 điểm)**

Bài: Viết chương trình mã hóa bằng thuật toán A5/1

Ý tưởng làm bài: dùng ngôn ngữ C# và lập trình Window Forms App

**Thuật toán A5/1:**

**Desgin Demo:**



**Các thành phần trong chương trình:**

**A5/1**: là một dãy kí tự 0,1 (ngẫu nhiên) có độ dài **23**, được nhập từ người sử dụng

**PlaniText**: là một dãy kí tự 0,1 (ngẫu nhiên) có độ dài **tùy ý,** được nhập từ người sử dụng

**Create Key**: Nút Button, có tác dụng chuyển đổi từ Key A5/1 thành Key

**Key**: là nơi chứa đựng giá trị được tính từ Key A5/1 khi nhấn Nút button **Create Key**

**Result**: Nút Button, có tác dụng tính kết quả của quá trình mã hóa Thuật toán A5/1

**List Result**: là nơi chứa dựng các kết quả mã hóa khi nhấn Nút **button Result**

**Delete**: Nút button, có tác dụng xóa kết quả mà người dùng chọn ra khỏi **List Result**

**Exit**: Nút Button, có tác dụng thoát khỏi chương trình mã hóa.

**ReLoad**: Nút button, có tác dụng cập nhật lại danh sách kết quả sau khi tìm kiếm

**Search PlainText:** là dãy kí tự 0,1(ngẫu nhiên) có độ dài tùy í, mà người sử dụng muốn tìm

**Search**: là Nút Button có tác dụng tìm kím dãy kí tự **PlainText.**

**Undo**: là Nút Button có tác dụng trở về khi nhấn Nút button **Delete**

**Save:** Nút Button, có tác dụng lưu danh sách **List Result**

**List Data Save:** danh sách lưu trữ dữ liệu của **List Result**

**Code Build:**

🖝Tạo class chứa dữ liệu kết quả mã hóa:

|  |
| --- |
| internal class clssResult  {  private string m\_A5;  private string m\_PlainText;  private string m\_Key;  private string m\_Result;  public string A5 { get => m\_A5; set => m\_A5 = value; }  public string PlainText { get => m\_PlainText; set => m\_PlainText = value; }  public string Key { get => m\_Key; set => m\_Key = value; }  public string Result { get => m\_Result; set => m\_Result = value; }  public clssResult(string a5, string plainText, string key, string result)  {  A5 = a5;  PlainText = plainText;  Key = key;  Result = result;  }  } |

🖝Tạo danh sách chứa dữ liệu giải mã bằng Thuật Toán A5/1 đưa vào **List Result**:

|  |
| --- |
| // nơi giữ giá trị kết quả của Dữ liệu nhập  private List<clssResult> ClssResult = new List<clssResult>(); |

🖝Tạo danh sách chứa dữ liệu tìm kiếm bằng đưa vào **List Result**:

|  |
| --- |
| // nơi giữ giá trị dữ liệu được tìm kiếm  private List<clssResult> ClssSearch = new List<clssResult>(); |

🖝Tạo danh sách chứa dữ liệu được lưu trữ đưa vào **List Data Save**:

|  |
| --- |
| // nơi giữ giá trị Save  List<clssResult> Clss\_Save = new List<clssResult>(); |

🖝Các hàm hiển thị danh sách: ClssResult, ClssSearch, Clss\_Save

|  |
| --- |
| // hiển thị kết quả của dữ liệu nhập vào trong datagirdview "dgvResult"  private void show\_Result()  {  dgvResult.DataSource = ClssResult.ToList();  }  // hiển thị kết quả của dữ liệu được tìm kiếm từ dữ liệu nhập vào trong datagirdview "dgvResult"  private void show\_Search()  {  dgvResult.DataSource = ClssSearch.ToList();  }  //hiển thị kết quả của dữ liệu được lưu vào trong datagirdview "dgv\_Save"  private void show\_Save\_Data()  {  dgv\_Save.DataSource = Clss\_Save.ToList();  }  int nb\_Save = 0;// tác dụng cập nhật lần lưu trữ của dữ liệu nhập vào  bool Emtyp\_Result = true;// danh sách nhập vào rỗng  bool Check\_Save\_Click = false;//Có tác dụng kiểm tra việc lưu |

🖝 Hàm xử lí sự kiện với Nút button **Create Key**:

|  |
| --- |
| private void bntCreateKey\_Click(object sender, EventArgs e)  {  string Plaintext = txtPlainText.Text.Trim();//nhập PlainText  string TextA5 = txtA5.Text.Trim();//23 chu so [0,1]  string key = "";  if (TextA5.Length < 23)  {  MessageBox.Show("Chưa Đủ Kí Tự A5/1 [0,1]", "Thong bao", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);  return;  }  //vong lap co tac dung kiem tra 23 ki tu cua Key A5 gom [0,1] hay khong  foreach (char kitu in TextA5)  {  if (kitu != '0' && kitu != '1')  {  MessageBox.Show("Lỗi A5", "THÔNG BÁO", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);  return;  }  }  //vong lap co tac dung kiem tra ki tu cua PlainText gom [0,1] hay khong  foreach (char Kitu in Plaintext)  {  if (Kitu != '0' && Kitu != '1')  {  MessageBox.Show("Lỗi PlainText", "THÔNG BÁO", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);  return;  }  }  //tạo x, y, z chứa lần lượt 6,8,9 kí tự của A5  string x = TextA5.Substring(0, 6);  string y = TextA5.Substring(6, 8);  string z = TextA5.Substring(14, 9);  int i = 0;//Đếm số bit của PlanText  // Lập theo số bit PlainText, vd: PlainText = "111" -> 3 bit, lập quá trình 3 lần  while (i < Plaintext.Length)  {  // tao mảng chứa kí tự tại vị trí x[1], y[3],z[3]  string[] array = { x.Substring(1, 1), y.Substring(3, 1), z.Substring(3, 1) };  int dem = 0;  string Major = "0";  /\* Quy tắc: Tính Major  \* Theo quy tắc số đông khi so sánh các kí tự x[1], y[3], z[3] với nhau  \* vd: (0,0,1) -> Major = 0  \*/  for (int j = 0; j < array.Length; j++)// Vòng lập để đếm kí tự 0,1  {  if (array[j] == "0")//kiểm tra kí tự 0  {  dem++;  }  }  if (dem < 2)// nếu kí tự 0 ít hơn 2  {  Major = "1";  }  /\*Quy tắc: Tìm Key  \* giá trị tại vị trí x[1], y[3], z[3]  \* Nếu bằng Major thì chuyển kí tự cuối lên đầu chuỗi  \* Ngược lại giữ nguyên  \* Cập nhật lại chuỗi x, y, z  \* Lấy kí tự cuối cùng trong chuỗi x, y, z  \* Dùng phép XOR: các kí tự cuối cùng trong chuỗi x, y, z  \*/  if (x.Substring(1, 1) == Major)  {  StringBuilder sbx = new StringBuilder(x);  char kytucuoi = sbx[sbx.Length - 1];  sbx.Remove(sbx.Length - 1, 1);  sbx.Insert(0, kytucuoi);  x = sbx.ToString();  }  string Last\_x = x.Substring(5, 1);  if (y.Substring(3, 1) == Major)  {  StringBuilder sby = new StringBuilder(y);  char kytucuoi = sby[sby.Length - 1];  sby.Remove(sby.Length - 1, 1);  sby.Insert(0, kytucuoi);  y = sby.ToString();  }  string Last\_y = y.Substring(7, 1);  if (z.Substring(3, 1) == Major)  {  StringBuilder sbz = new StringBuilder(z);  char kytucuoi = sbz[sbz.Length - 1];  sbz.Remove(sbz.Length - 1, 1);  sbz.Insert(0, kytucuoi);  z = sbz.ToString();  }  string Last\_z = z.Substring(8, 1);  // mảng chứa các kí tự cuối cùng của x,y,z  string[] Lastchar = { Last\_x, Last\_y, Last\_z };  int tam = 0;  // Vòng lập xét qua các kí tự cuối cùng của x,y,z  for (int j = 0; j < Lastchar.Length; j++)  {  if (Lastchar[j] == "1")  {  tam++;// đếm kí tự 1  }  }  string kq = "1";  if (tam == 0 || tam == 2)// trường hợp này là không có kí tự 1 hoặc có 3 kí tự 1  {  kq = "0";// qui tắc cộng nhị phân  }  key += kq;// kq được thêm vào chuỗi result  i++;//biến tăng  }  //Thoát vòng lập, kết quả được hiển thị tại txtKey  txtKey.Text = key;  } |

* Xuất thông báo “Lỗi” khi nhập sai dữ liệu đầu vào

🖝 Hàm kiểm tra dữ liệu nhập vào:

|  |
| --- |
| private bool check\_Insert()  {  string check\_A5 = txtA5.Text;  string check\_PlainText = txtPlainText.Text;  for (int i = 0; i < ClssResult.Count; i++)  {  if (check\_A5 == ClssResult[i].A5 && check\_PlainText == ClssResult[i].PlainText)  {  MessageBox.Show("Đã Có Dữ Liệu", "THÔNG BÁO", MessageBoxButtons.OK);  txtA5.Focus();  Clear\_Text();  return false;  }  }  return true;  } |

🖝 Hàm xử lí sự kiện với Nút button Result:

|  |
| --- |
| private void bntResult\_Click(object sender, EventArgs e)  {  if (txtKey.Text == "")  {  MessageBox.Show("Chưa Tạo Key", "THÔNG BÁO", MessageBoxButtons.OK);  return;  }  //dữ liệu nhập đã được cho vào datagirdview "dgvResult"  Emtyp\_Result = false;//không rỗng  //dữ liệu mới nhập vào chưa được lưu  Check\_Save\_Click = false;  //so sanh du lieu co trong datalisview  if (!check\_Insert())  return;  //biến được gán kết quả thuật toán mã hóa A5/1  string Result = "";  /\*Quy tắc tính kết quả của thuật toán mã hóa A5/1  \*Phép XOR giữa PlainText và Key  \*vd: PlainText = "111" ; Key = "010" -> kết quả = "101"  \*/  for (int i = 0; i < txtKey.Text.Length; i++)  {  if (txtPlainText.Text[i] == txtKey.Text[i])  {  Result += "0";  }  else  {  Result += "1";  }  }  //truyền dữ liệu vào datagirdview  clssResult bs = new clssResult(txtA5.Text, txtPlainText.Text, txtKey.Text, Result);  ClssResult.Add(bs);  show\_Result();  Clear\_Text();  } |

🖝 Hàm trả về các giá trị nhập về rỗng:

|  |
| --- |
| private void Clear\_Text()  {  txtA5.Text = "";  txtPlainText.Text = "";  txtKey.Text = "";  txtA5.Text = "";  } |

🖝 Hàm xử lí sự kiện với Nút button Delete:

|  |
| --- |
| private void bntDelete\_Click(object sender, EventArgs e)  {  int index;  if (Check\_Save\_Click)// dữ liệu đã được lưu  {  if (dgvResult.SelectedRows.Count > 0)//sự kiện chọn dòng  {  // Xóa dòng khỏi DataGridView  index = dgvResult.SelectedRows[0].Index;  MessageBox.Show("Xóa Dữ Liệu Này?", "THÔNG BÁO", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);  dgvResult.Rows.RemoveAt(index);  dgv\_Save.Rows.RemoveAt(index);  //thực hiện xóa -> dữ liệu chưa lưu  Check\_Save\_Click = false;  nb\_Save--;  }  else// không chọn dòng nào  MessageBox.Show("Vui lòng chọn một dòng để xóa.", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  else//chưa lưu Dữ liệu  {  if (dgvResult.SelectedRows.Count > 0)  {  index = dgvResult.SelectedRows[0].Index;  MessageBox.Show("Xóa Dữ Liệu Này?", "THÔNG BÁO", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);  dgvResult.Rows.RemoveAt(index);  }  else  MessageBox.Show("Vui lòng chọn một dòng để xóa.", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  } |

🖝 Hàm xử lí sự kiện với Nút button Save:

|  |
| --- |
| private void bntSave\_Click(object sender, EventArgs e)  {  string Save\_A5 = "";  string Save\_PlainText = "";  string Save\_Key = "";  string Save\_Result = "";  while (nb\_Save < ClssResult.Count)  {  Save\_A5 = ClssResult[nb\_Save].A5;  Save\_PlainText = ClssResult[nb\_Save].PlainText;  Save\_Key = ClssResult[nb\_Save].Key;  Save\_Result = ClssResult[nb\_Save].Result;  clssResult bs\_Sa = new clssResult(Save\_A5, Save\_PlainText, Save\_Key, Save\_Result);  Clss\_Save.Add(bs\_Sa);  show\_Save\_Data();  nb\_Save++;  }  Check\_Save\_Click = true;//Nút Save được Click  MessageBox.Show("Đã Lưu", "THÔNG BÁO", MessageBoxButtons.OK);  } |

🖝 Hàm xử lí sự kiện với Nút button Exit:

|  |
| --- |
| private void bntExit\_Click(object sender, EventArgs e)  {  //Chưa nhấn Nút Save  if (Emtyp\_Result == true || Check\_Save\_Click == true)  {  Close();  }  else  MessageBox.Show("Dữ Liệu Chưa Được Lưu", "THÔNG BÁO", MessageBoxButtons.OK);  } |

🖝 Hàm xử lí sự kiện với Nút button Search:

|  |
| --- |
| private void bntSearch\_Click(object sender, EventArgs e)  {  int number = 0;  if (!check\_Searches())  {  return;  }  for (int i = 0; i < ClssResult.Count; i++)  {  if (txtSearchPlText.Text == ClssResult[i].PlainText)  {  string S\_plainText = ClssResult[i].PlainText;  string S\_A5 = ClssResult[i].A5;  string S\_Key = ClssResult[i].Key;  string S\_Result = ClssResult[i].Result;  clssResult bs\_sr = new clssResult(S\_A5, S\_plainText, S\_Key, S\_Result);  ClssSearch.Add(bs\_sr);  check\_Search = true;  }  else  number++;  }  if (number == ClssResult.Count)  {  MessageBox.Show("KHÔNG CÓ DỮ LIỆU", "Thông Báo");  return;  }  show\_Search();// hiển thị giá trị tìm kiếm được  } |

🖝 Hàm xử lí sự kiện với Nút button ReLoad:

|  |
| --- |
| private void button1\_Click(object sender, EventArgs e)  {  ClssSearch.Clear();  show\_Result();// hien thi danh sach ket qua tong.  } |

**Mô tả Chương trình hoạt động:**

Key A5: 100101 01001110 100110000

PlainText: 111

* Từ Key A5 tách thành 3 chuỗi con x,y,z chứa giá trị của Key A5

x = 100101 (6 kí tự)

y = 01001110 (8 kí tự)

z = 100110000 (9 kí tự)

Quá trình tìm giá trị Key: PlainText là 111 (3 bit) -> lập lại quá trình 3 lần

* Tính Major = (x[1], y[3], z[3]); {1, 3, 3} vị trí kí tự của x,y,z

Major = (0,0,1) = 0; theo qui tắc số đông giữa các kí tự, ta được Major = 0;

* So sánh x[1], y[3], z[3] với Major = 0;
* Nếu kí tự của x[1], y[3], z[3] bằng với Major thì di chuyển kí tự cuối cùng của [x, y, z] lên vị trí đầu tiên, ngược lại giữ nguyên chuỗi; Cập nhật lại chuỗi [x, y, z] (I)

x[1] (0) == Major (0) -> x = 1100010

y[3] (0) == Major (0) -> y = 00100111

z[3] (1) != Major (0) -> z = 100110000

* Lấy kí tự cuối cùng của chuỗi [x,y,z] vừa cập nhật
* Dùng phép OXR -> 0 + 1 + 0 = 1
* Lấy kết quả vừa có ( = 1) gán vào Key.
* Ta được giá trị kí tự đầu tiên của Key = 1;
* Sau đó, thực hiện lại Quá trình tìm Key từ chuỗi [x, y ,z] (I)
* Kết quả tiếp theo lần lượt là: 0, 0 -> Key = 100;

Quá trình Cipher:

* Sử dụng phép OXR: PlainText (111) với Key (100) - > Cipher = 011;

**Hướng dẫn thực thi chương trình:**

Các sự kiện cần thiết:

B1: Nhập **A5** (độ dài 23) và **Plain Text** (độ dài tùy ý), đều chứa kí tự [0,1]

B2: Nhấn button **Create Key** < **private void bntCreateKey\_Click(object sender, EventArgs e**) >: được thực thi -> hiển thị kết quả ở **Key**

B3: Nhấn button Result < private void bntResult\_Click(object sender, EventArgs e)> được thực thi -> hiện thị kết quả ở List Result.

B4: Nhấn button **Save**, để lưu **List Result vào List Data Save**

B5: Nhấn button **Exit**, thoát khỏi chương trình.

Hết