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 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 k_1, k_2, k_3, … 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)`
	`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);`
	`  }`
	`}`

```
		{
			create_Node(t->right, x);
		}
		else
			cout << "\nTon tai Node";
	}
}
```

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

```
int Node_Max(Tree_BST t)
{
	if (t->left == NULL && t->right
== NULL)
	{
		return t->data;
	}
	return Node_Max(t->right);
}
```

Node_Min: Tìm Nút có giá trị nhỏ nhất

```
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

```
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
```

Remove_BST: Hủy Cây

```
void Remove_BST(Tree_BST& t)
{
	if (t == NULL)
		return;
	Remove_BST(t->left);
	Remove_BST(t->right);
	delete t;
	t = NULL;
}
```

```
            {
                  LeftMostChild(X, t->right);
            }
            delete X;
      }
}
```

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

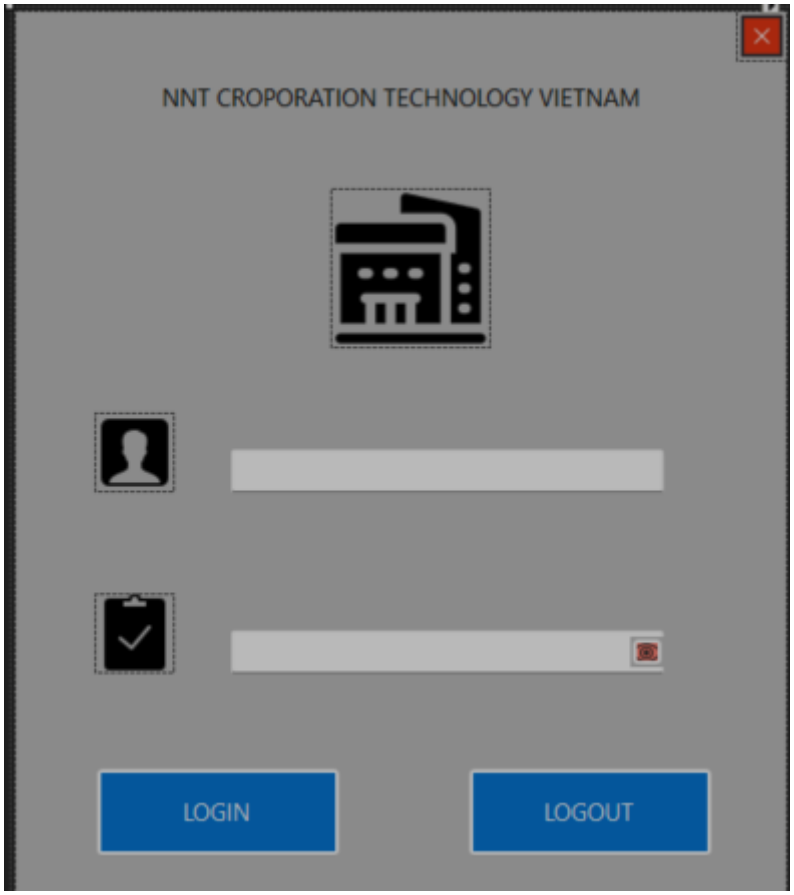
+ Sử dụng cấu trúc dữ liệu phù hợp để lưu trữ thông tin nhân viên thực hiện các yêu cầu sau:

a. Thêm/ xóa/ sửa thông tin của nhân viên
b. Cho biết nhân viên nào có lương cao nhất, nhân viên nào có lương thấp nhất
c. 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”);

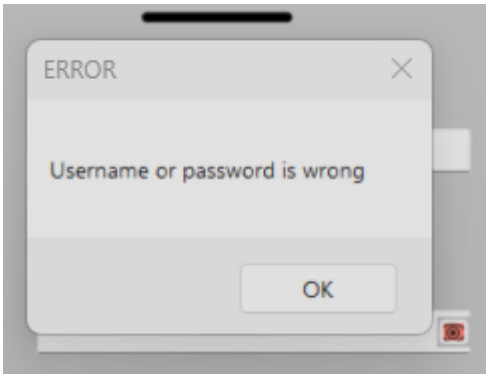
txtPasswork: Nhập mật khẩu; hiện trên màn hình là dấu * (Password: “123”)

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:

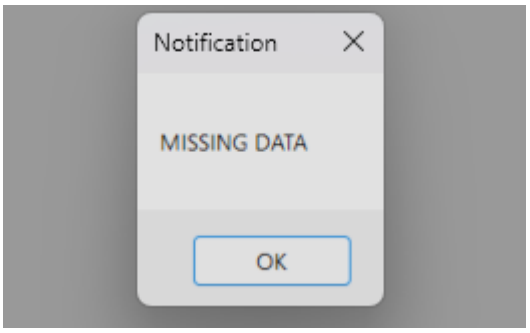
Button INSERT:

```
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();
```

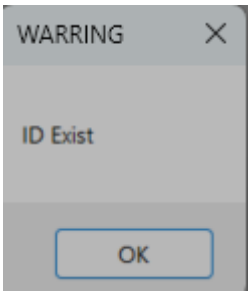
```
        clear_Text();
        Check_Save_Click = false;// chưa lưu
        Empty_Data = false;//danh sach khong rong

    }
```

+ 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

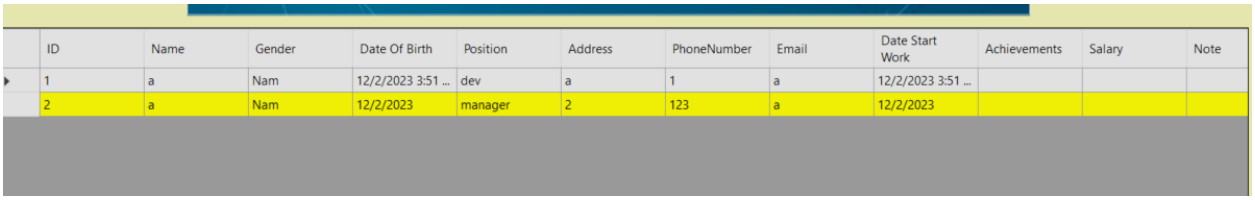
```
        public C_InfoEmployee check_ID(string id)// kiểm tra ID
        {
            foreach (C_InfoEmployee Employee in C_common.clss_Infor)
            {
                if (Employee.ID == id)
                    return Employee;
            }
            return null;
        }
```

Button SEARCH: Tìm kiếm theo ID

```
private void bntSearch_Click(object sender, EventArgs e)//Tìm ID
{
```

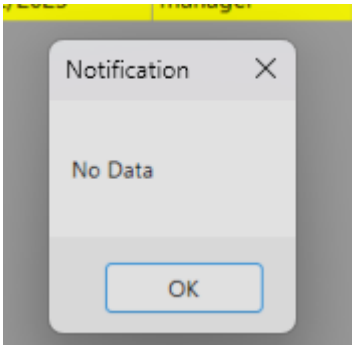
```
    C_InfoEmployee search_ID = check_ID(txtSearchID.Text);
    if (search_ID == null)
    {
        MessageBox.Show("No Data", "Notification");
        txtSearchID.Clear();
        return;
    }

    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



ID	Name	Gender	Date Of Birth	Position	Address	PhoneNumber	Email	Date Start Work	Achievements	Salary	Note
1	a	Nam	12/2/2023 3:51 ...	dev	a	1	a	12/2/2023 3:51 ...			
2	a	Nam	12/2/2023	manager	2	123	a	12/2/2023			

+ 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;
             }
         }
```

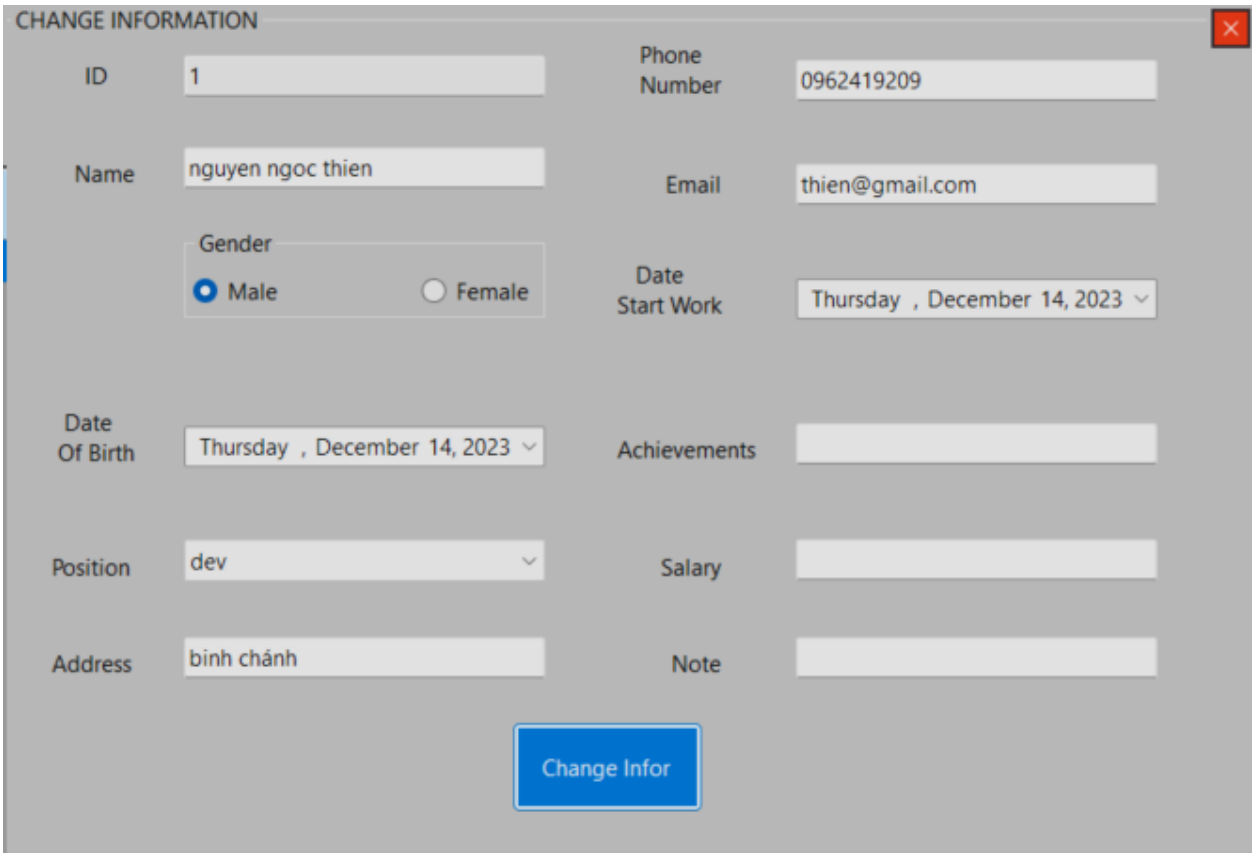
Button CHANGE INF:

```
private void bntfix_Click(object sender, EventArgs e)
 {
     if (choseRow)
     {
         changeInformation.ShowDialog();
         show_list_infor_Employee();
         DecorColor(rowIndex);
```

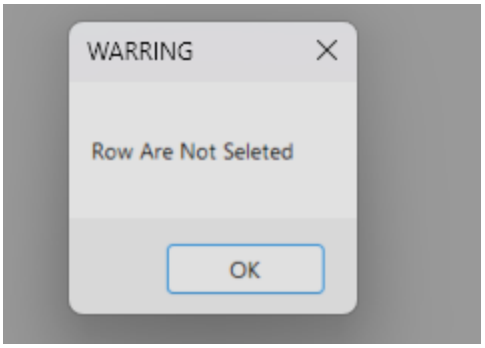
```
            choseRow = false;
            Empty_Data = false;
            Check_Save_Click = false;
        }
        else
            MessageBox.Show("Row Are Not Seleted", "WARRING");
    }
```

+ 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:

ID	Name	Gender	Date Of Birth	Position	Address	PhoneNumber	Email	Date Start Work	Achievements	Salary	Note
1	A	Nam	12/2/2023 4:08 ...	dev	a	12	a	12/2/2023 4:08 ...			

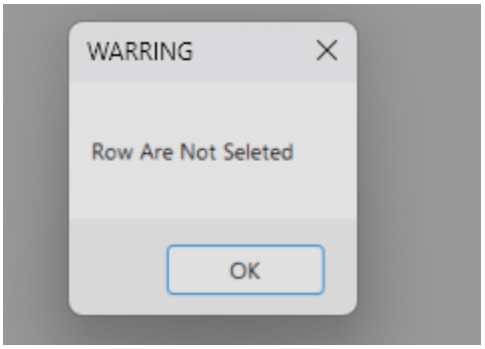
Dữ liệu được thay đổi:

ID	Name	Gender	Date Of Birth	Position	Address	PhoneNumber	Email	Date Start Work	Achievements	Salary	Note
1	Nguyễn Ngọc T...	Nam	4/20/2003 4:08 ...	dev	Bình Chánh	0962419209	nguyenngocthie...	12/2/2023 4:08 ...			

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"

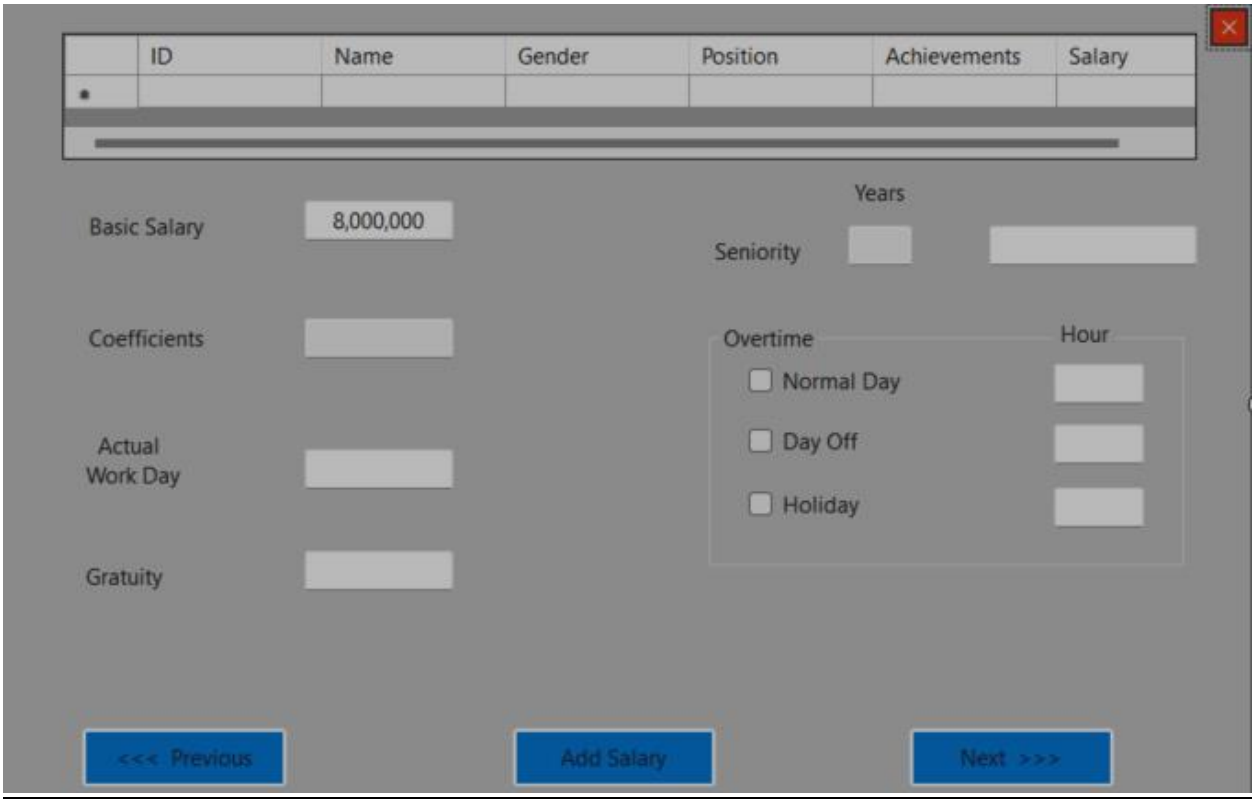
```
private void bntOpenFile_Click(object sender, EventArgs e)
 {
     try
     {
         OpenFileDialog openfile = new OpenFileDialog();
         //Thiết lập hộp thoại mở tệp
         openfile.Title = "Chọn tệp tin";
         openfile.Filter = "Tệp tin văn bản|*.txt|Tất cả các tệp tin|*.*";
         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.

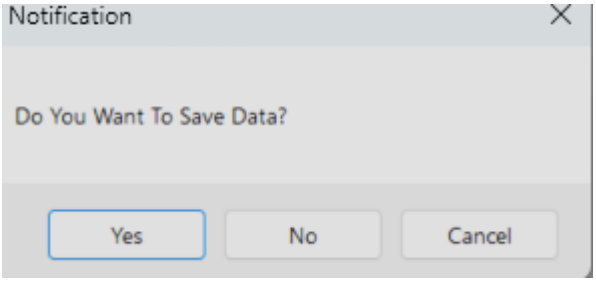
Button Previous: chuyển đến nhân viên trước đó

```
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

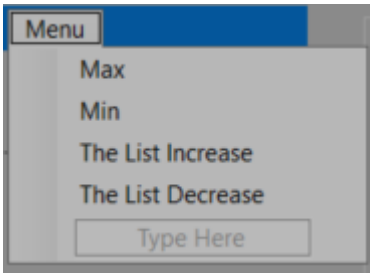
```
private void bntExit_Click(object sender, EventArgs e)//Thoát khỏi
frmInfoEmployee, trở về frmLogin
{
    if (Check_Save_Click == false && Empty_Data == false)
    {
        var a = MessageBox.Show("Do You Want To Save Data?", "Notification",
MessageBoxButtons.YesNoCancel);
        if (a == DialogResult.Cancel)
        {
            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();
 }
```

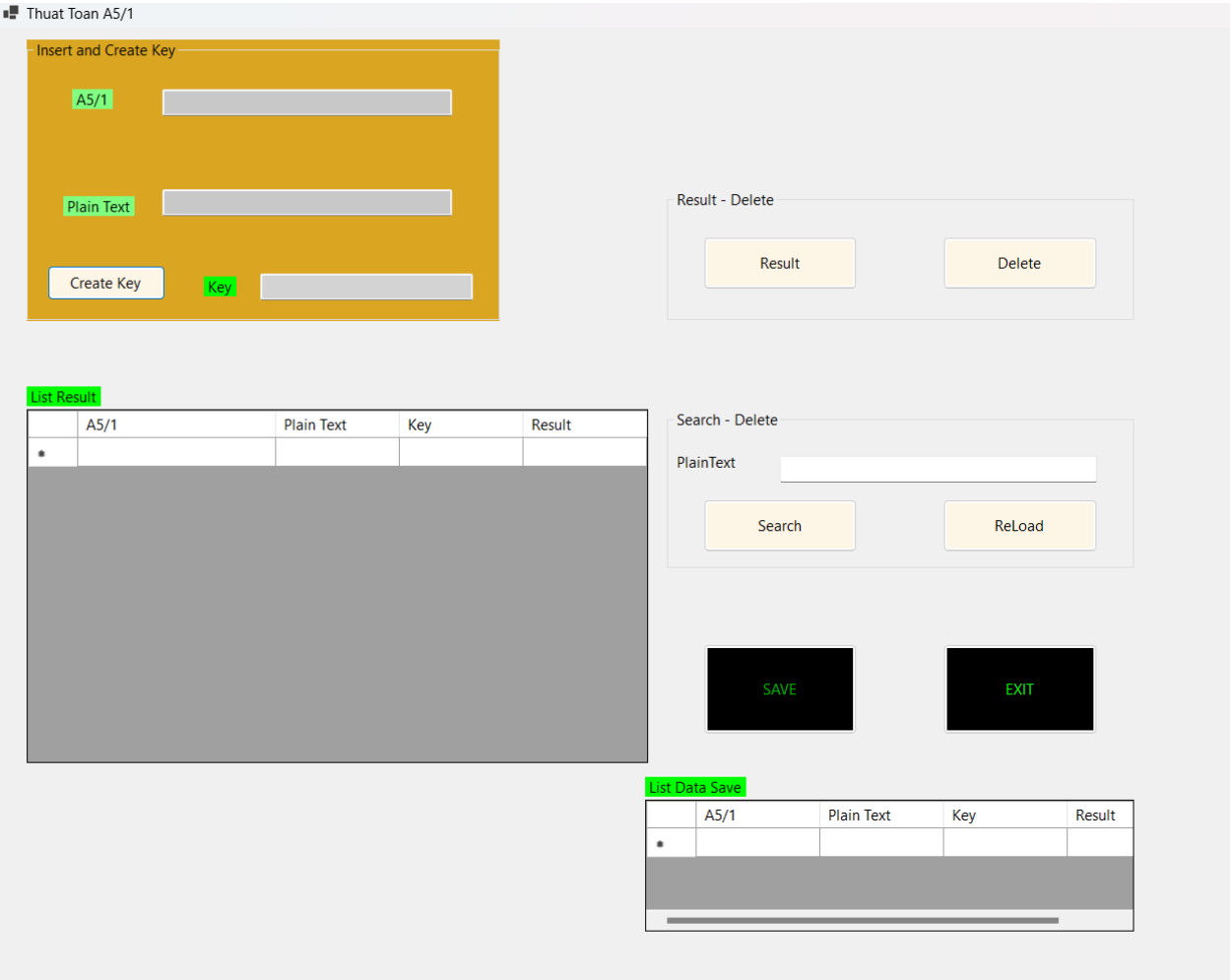
Phần III: Phần Mở Rộng (1.0 điểm)

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