

Lab 01:

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG C#

Với ứng dụng CONSOLE

A. MỤC TIÊU:

- ✓ Hướng dẫn sinh viên làm quen với ngôn ngữ lập trình C#: qua việc viết các ứng dụng **Console** trong vs. NET 2015/2017/ 2019.
- ✓ Xây dựng các lớp, tạo đối tượng, truy xuất các phương thức, ...
- ✓ Soạn thảo mã nguồn, biên dịch, debug, thực thi chương trình...
- ✓ Kế thừa trong lập trình hướng đối tượng trên C#.
- ✓ Tìm hiểu về sử dụng **lambda expression** trong thư viện **LINQ** của .NET
- ✓ Khuyến khích sinh viên sử dụng chuẩn viết code C# (**C# Coding Convetion**)
- ✓ Ngôn ngữ C#

- Cấu trúc chương trình C#: Namespace, class (thuộc tính, phương thức), Comment
- Kiểu dữ liệu: `bool`, `decimal`, `double`, `float`, `int`, `string`, `DateTime`
`bool?`, `decimal?`, `double?`, `float?`, `int?`, `long?`, `DateTime?`...
`Nullable<bool>`, `Nullable<decimal>`, `Nullable<double>`, `Nullable<float>`, `Nullable<int>`...
`object`, `var`, `dynamic`
- Chuyển đổi kiểu dữ liệu: Ép kiểu (`int`), `as`, `Parse`, `Convert`
- Vòng lặp: `for`, `foreach`, `while`, `do.. while` và Lệnh điều khiển `break`, `continue`, `return`
- Tính đóng gói: bởi sử dụng Access Specifier: `private`, `protected`, `internal`, `protected internal`, `public`
- Tính kế thừa: để tái sử dụng code giúp thời gian thực thi nhanh hơn
`class <derived_class> : <base_class>`
- Coding Convention C#: Đưa ra các quy ước khi coding với ngôn ngữ lập trình C#, với các quy tắc này giúp tiết kiệm thời gian rất trong quá trình phát triển và bảo trì phần mềm.
 Ví dụ: Đặt tên class dùng danh từ, đối với phương thức dùng động từ ...
 Tên biến, tên phương thức thể hiện được ý nghĩa
 Nên comment những đoạn code khó hiểu hoặc có chức năng đặc biệt
- Một số phương thức của thư viện Console
`Console.Write(<giá trị cần in ra màn hình>);`
`Console.WriteLine();` // Sử dụng lệnh in ra màn hình có xuống dòng
`Console.Read();` //đọc 1 ký tự và trả về 1 số nguyên là mã ASCII của ký tự đó
`Console.ReadLine();` // Đọc dữ liệu chuỗi từ bàn phím cho đến khi gặp ký tự xuống dòng
`Console.ReadKey();` // Dừng màn hình để xem kết quả.

B. BÀI TẬP

Bài tập 1A: Tạo lớp Student có các dữ liệu và phương thức sau

- Mã số sinh viên
- Họ Tên sinh viên
- Điểm trung bình học tập
- Khoa
 - + Tạo Property cho các thành viên trên
 - + Tạo constructor không có tham số, và có tham số

Viết chương trình ở hàm **Main** () cho phép người dùng nhập vào số **N** là tổng số sinh viên, sau đó lần lượt tạo các đối tượng sinh viên và đưa vào mảng **Student** theo những thông tin do user nhập vào (dùng vòng lặp for). Cuối cùng xuất ra danh sách chi tiết thông tin sinh viên.

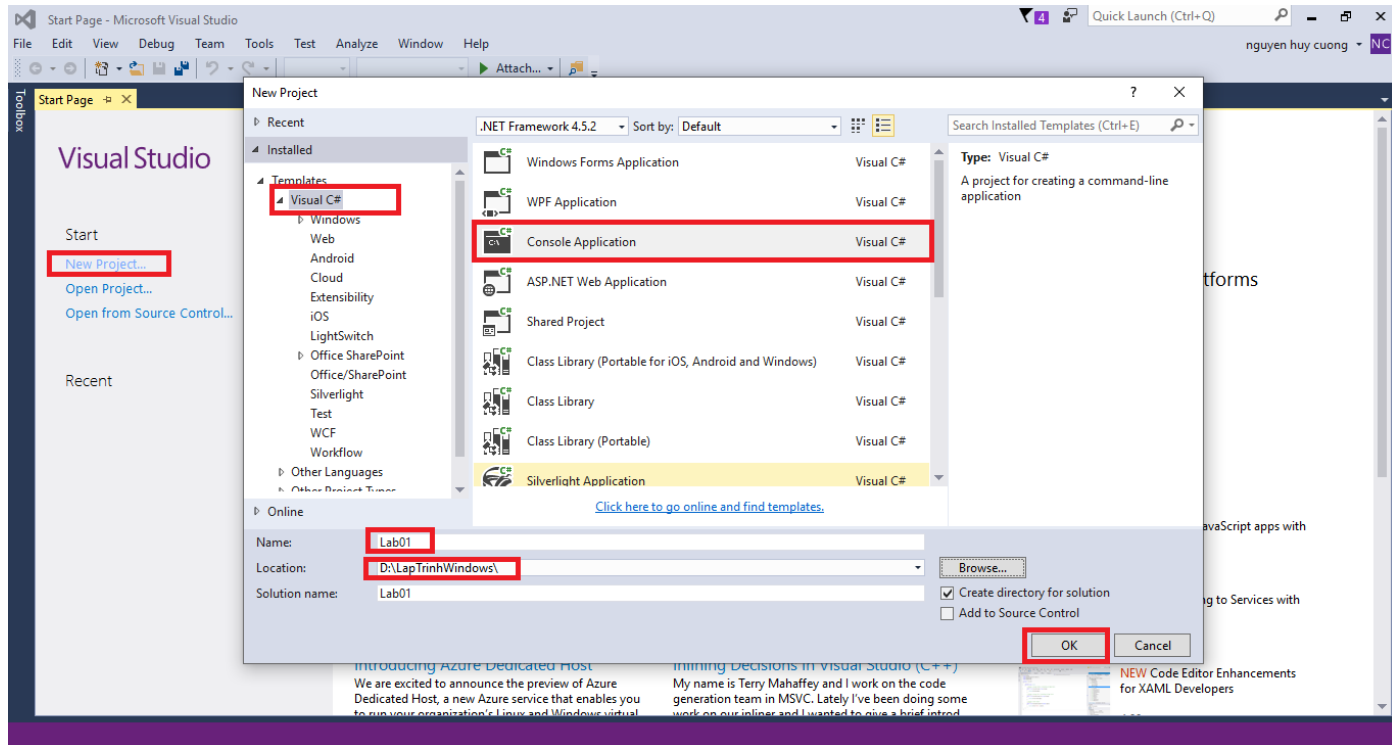
Yêu cầu:

- ✓ Sinh viên xây dựng chương trình theo nội dung mô tả bên trên.
- ✓ Compile & Build chương trình.
- ✓ Run chương trình ở hai chế độ debug và không debug.
- ✓ Chạy từng bước chương trình trong chế độ debug: dùng breakpoint hoặc chạy từng dòng lệnh. Kiểm tra những giá trị của các biến trong chương trình ở cửa sổ Watch.

Hướng dẫn:

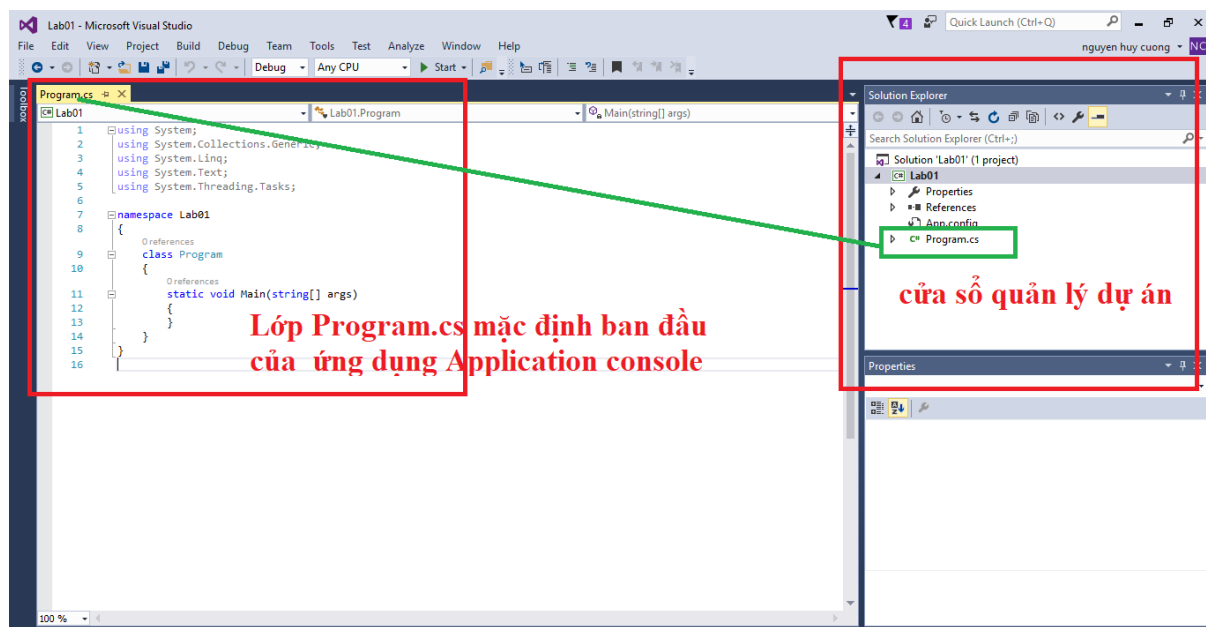
Bước 1: Tạo project mới trong VS 2015

- Mở Visual Studio 2015, chọn **New Project**, Chọn ngôn ngữ **Visual C#**, Loại project là **Console Application**
- Đặt tên project là **Lab01** và lưu



Hình 1: Tạo một project console c# mới trong VS .NET 2015

- Click OK để đồng ý tạo project, kết quả chúng ta được một ứng dụng console như sau:



Hình 2: Màn hình làm việc của Project

- Lưu lại dự Án File/Save All

Bước 2: Viết yêu cầu tạo class Student, Khai báo thuộc tính, phương thức, SV tự làm hoàn chỉnh theo yêu cầu của bài tập 1

```
class Student
{
    //1. Tao thuoc tinh
    private string studentID; //mã số sinh viên
}
```

```
private string fullName;    // họ tên sinh viên
private float averageScore; //điểm trung bình
private string faculty;     //khoa

//2. Tao cac Property
public string StudentID
{
    get
    {
        return studentID;
    }
    set
    {
        studentID = value;
    }
}
// 3. Tao constructor không tham số
public Student()
{
}
// 4. Tao constuctor có tham số
public Student(string id, string name, float score, string faculty)
{
    studentID = id;
    //....
}
}
```

Bước 3: Viết code trong hàm **Main()** để cho phép người dùng nhập vào tổng số sinh viên

```
static void Main(string[] args)
{
    //Nhap tong so sinh vien, Convert kiểu dữ liệu sang biến N
    kiểu int
    Console.WriteLine("Nhap tong so sinh vien N =");
    int N = Convert.ToInt32(Console.ReadLine());
    Student[] arrStudents = new Student[N];

    Console.WriteLine("\n ====NHAP DS SINH VIEN====");
    for (int i = 0; i < N; i++) //Lap n lan nhap thông tin sv
    {
        arrStudents[i] = new Student();
        Console.WriteLine("Nhap MaSV {0}:", i + 1);
        arrStudents[i].StudentID = Console.ReadLine();

        //SV lam tuong tu cho cac thông tin can nhap
    }

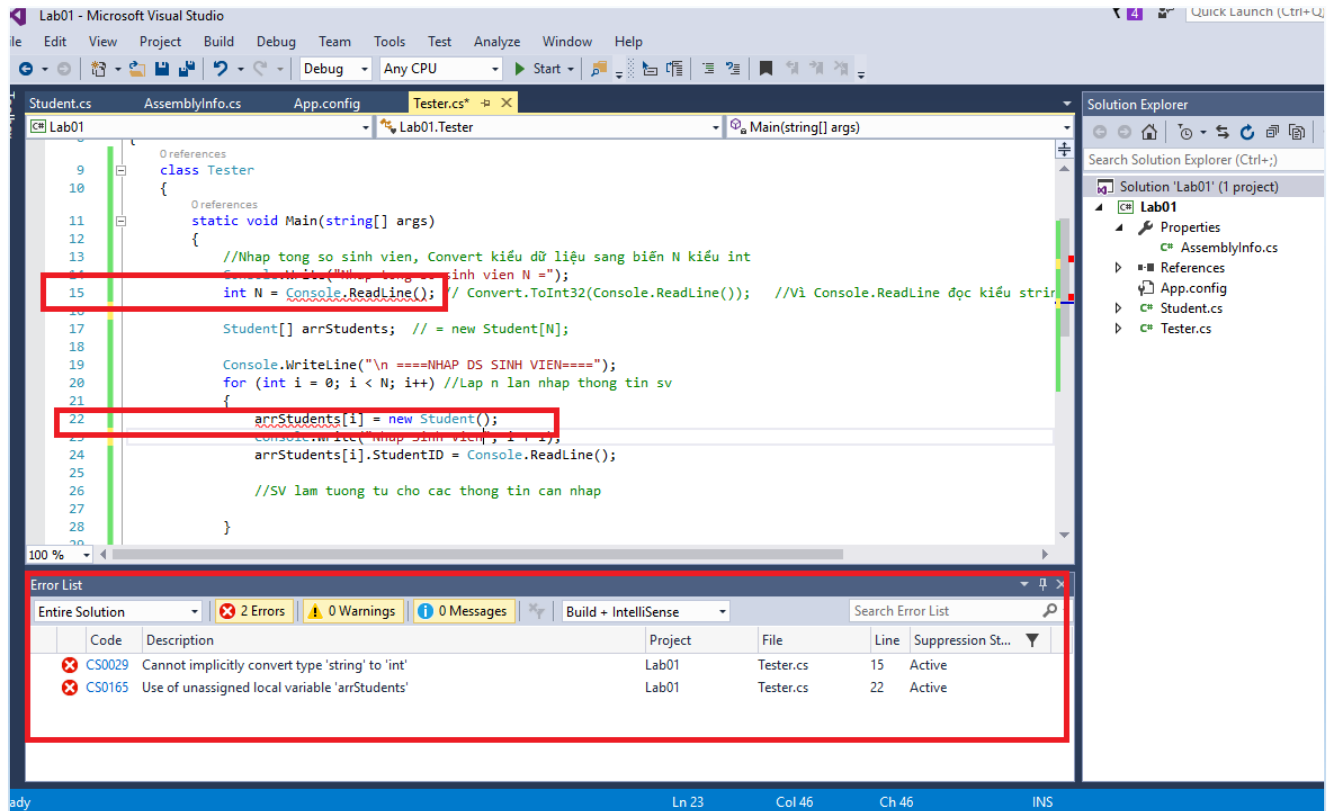
    Console.WriteLine("\n ====XUAT DS SINH VIEN====");
    foreach (Student sv in arrStudents)
    {
        Console.WriteLine("MSSV:{0}", sv.StudentID);
    }

    //De dung màn hình cho dễ kiểm tra kq
    Console.ReadKey();
}
```

Bước 4: Biên dịch và chạy chương trình

- ✓ Để biên dịch chương trình chọn menu **Build**, rồi chọn **Build Solution** (hoặc dùng phím tắt

F6 hoặc click phải vào Solution Explorer chọn **Build**). VS.NET sẽ thông báo biên dịch thành công hay gặp lỗi cú pháp.



Hình 4: VS thông báo tổng số lỗi, số dòng bị lỗi khi viết code

- ✓ Để chạy chương trình click vào biểu tượng **Start** trên VS (hoặc chọn **Debug / Debug Without Debugging** hoặc dùng **Ctrl +F5**)

```
C:\WINDOWS\system32\cmd.exe

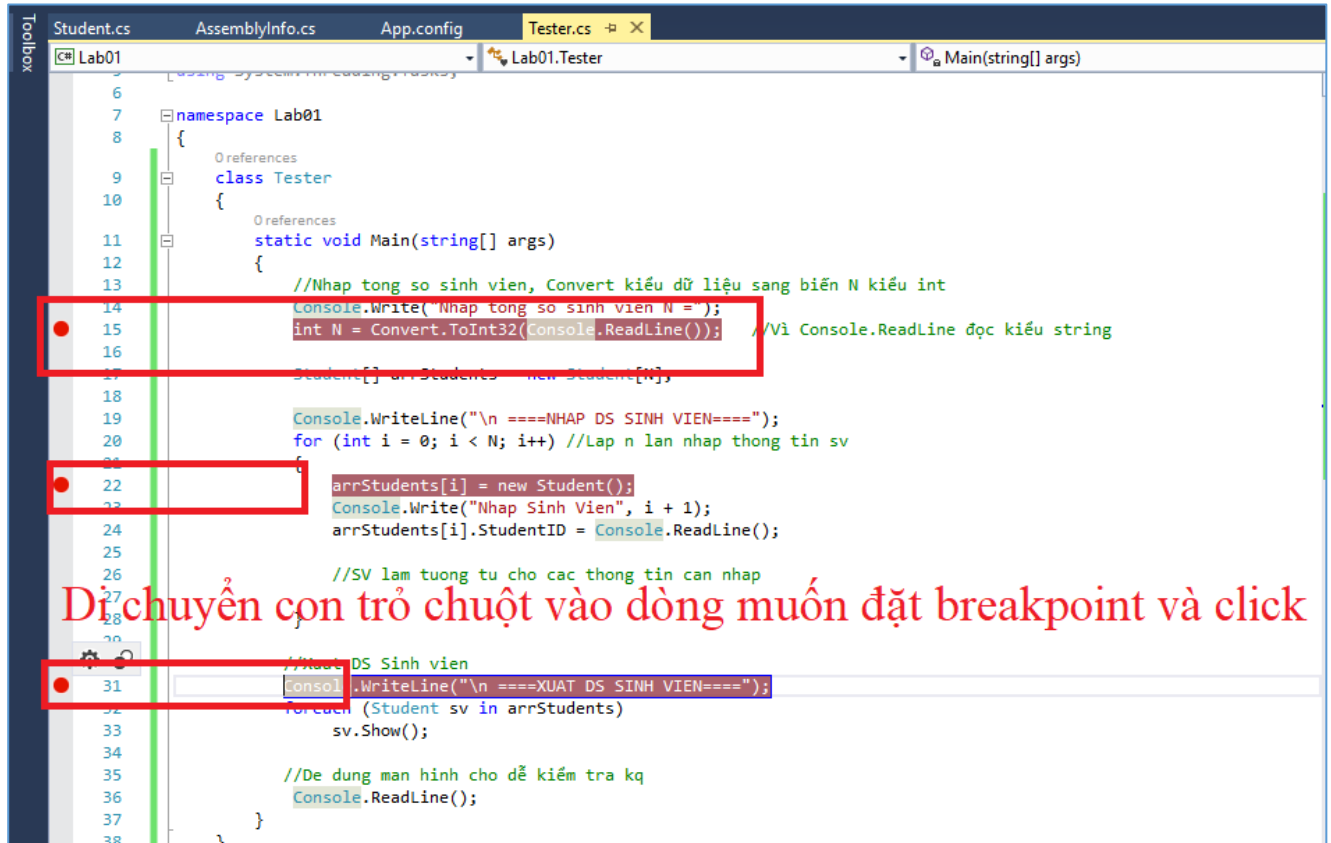
Nhap tong so sinh vien N =2

====NHAP DS SINH VIEN====
Nhap MaSV 1:CNMT0201001
Nhap MaSV 2:QTKD0293542

====XUAT DS SINH VIEN====
MSSV:CNMT0201001
MSSV:QTKD0293542
```

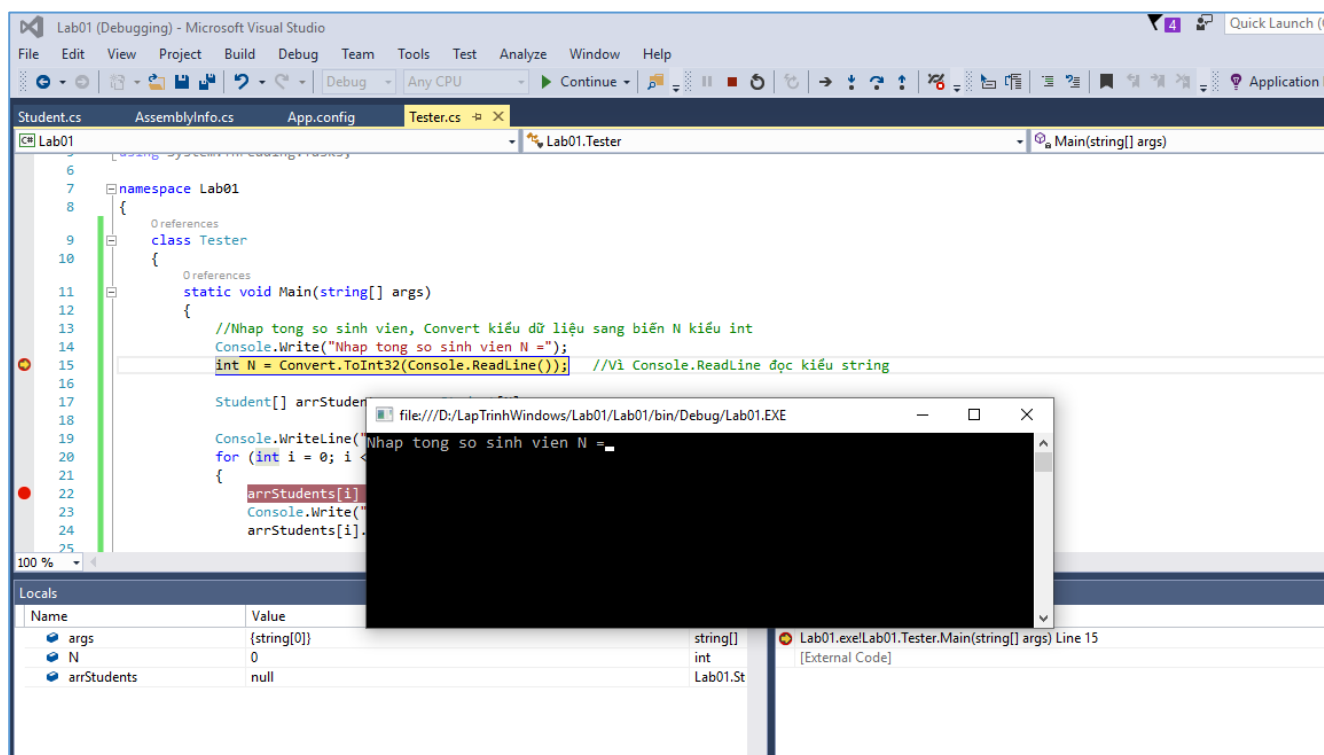
Hình 5: Kết quả màn hình console chương trình khi thực hiện

- ✓ Để bebug chương trình sử dụng F5(hoặc vào **Debug / Start Debugging**) kết hợp với việc đặt các breakpoint
 - Tạo breakpoint bằng cách đơn giản nhất là click chuột vào đầu dòng code (như trong hình). Để huỷ breakpoint, chỉ cần click chuột vào breakpoint đó một lần nữa. Ngoài ra các bạn cũng có thể tạo/huỷ breakpoint bằng phím F9.



Hình 6: Đặt các breakpoint hỗ trợ quá trình Debug

- Nhấn F5 để bắt đầu Debug, Tại các vị trí BreakPoint chương trình sẽ dừng lại. Muốn kiểm tra được sự thay đổi giá trị của các biến.



- Sử dụng Step Over / Step Into / Step Out trong quá trình Debug chương trình

Step Over (F10): Chạy step by step, lướt qua hàm (chỉ nhận giá trị return của hàm).

Step Into (F11): Chạy step by step, đi vào nội dung của các hàm con.

Step Out (continue): “Nhảy” đến breakpoint kế tiếp. Nếu không còn breakpoint nào thì sẽ kết thúc debug. Ngoài ra nó còn có chức năng chạy lướt qua hàm con hiện tại.

Bài tập 1B:

- ✓ Không viết lệnh nhập và xuất danh sách sinh viên trực tiếp trong hàm **Main()** mà hãy viết hai phương thức nhập và xuất thông tin sinh viên ở lớp **Student**.
- ✓ Sử dụng collection là **List** để chứa danh sách sinh viên thay thế cho mảng sinh viên trong bài tập 1. Chạy lại chương trình theo yêu cầu trên.

```
//Bước 1: Ở lớp Student thêm 2 phương thức Input và Show
public void Input()
{
    Console.Write("Nhập MSSV:");
    StudentID = Console.ReadLine();

    Console.Write("Nhập Điểm:");
    AverageScore = float.Parse(Console.ReadLine()); //ép sang kiểu float
    // Lam tuong tu cho cac thuoc tinh khac
}

public void Show()
{
    Console.WriteLine("MSSV:{0} Họ Tên:{1} Khoa:{2} ĐiểmTB:{3}",
this.StudentID, this.fullName, this.Faculty, this.AverageScore);
}

//Bước 2: Sửa lại ở Hàm Main, xóa mảng arrStudents sử dụng list
List<Student> listStudent = new List<Student>();
Console.WriteLine("\n ====NHAP DS SINH VIEN====");
for (int i = 0; i < N; i++) //Lap n lan nhap thong tin sv
{
    Student temp = new Student();
    temp.Input();
    listStudent.Add(temp); //đưa đối tượng temp vào List
}

Console.WriteLine("\n ====XUAT DS SINH VIEN====");
foreach (Student sv in listStudent)
{
    sv.Show();
}
```

- ✓ Viết tiếp chương trình thực hiện một số yêu cầu sau. (Khuyến khích sinh viên viết ra các hàm cho dễ quản lý, những phần chung nên gom vào một hàm)
- Xuất ra thông tin của các SV đều thuộc khoa “CNTT” (nếu có)
- Xuất ra danh sách các sinh viên theo thứ tự có điểm trung bình tăng dần.
- Xuất ra danh sách các Sinh viên có điểm trung bình cao nhất và thuộc khoa “CNTT” nếu có.

Hướng Dẫn Sử dụng Lambda expression trong thư viện LINQ

- Cú pháp của câu dạng truy vấn hỗ trợ một vài từ khóa đơn giản (Where, OrderBy, GroupBy, Max, Min, Sum, Average, Count...)

- Ví dụ

```
//Lấy DS Sinh Viên thuộc khoa CNTT
List<Student> listSVKhoaCNTT = listStudent.Where(p => p.Faculty ==
"CNTT").ToList();

//Lấy điểm TB cao nhất trong list listStudent
float maxScore = listStudent.Max(p => p.AverageScore);

//Sắp xếp DS Sinh Viên có điểm trung bình tăng dần
List<Student> listSortStudent = listStudent.OrderBy(p =>
p.AverageScore).ToList();

//Lấy ra 1 Sinh Viên bất kì đầu tiên có mã số là "CNTT020120"
Student find1 = listStudent.FirstOrDefault(p => p.StudentID == "CNTT020120");
Student find2 = listStudent.First(p => p.StudentID == "CNTT020120");

//Đếm tổng số lượng sinh viên có điểm < 5 và thuộc khoa CNTT
int totalStudent = listStudent.Where(p => p.AverageScore < 5 && p.Faculty ==
"CNTT").Count();
```

Hướng dẫn:- Để dễ dàng quản lý code sinh viên nên gom về các hàm (hàm nhập trả về List các sinh viên, Hàm xuất với tham số truyền vào List)

```
1 reference
private static List<Student> LayListSinhVienTuBanPhim()
{
    List<Student> listStudent = new List<Student>();
    Console.Write("Nhap tong so sinh vien N =");
    int N = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("\n ====NHAP DS SINH VIEN====");
    for (int i = 0; i < N; i++)
    {
        Student temp = new Student();
        temp.Input();
        listStudent.Add(temp); //đưa đối tượng temp vào List
    }
    return listStudent;
}

1 reference
private static void XuatDSSinhVien(List<Student> listStudent)
{
    Console.WriteLine("\n ====XUAT DS SINH VIEN===="); //Xuat DS Sinh vien
    foreach (Student sv in listStudent)
    {
        sv.Show();
    }
}

0 references
static void Main(string[] args)
{
    List<Student> listStudent = LayListSinhVienTuBanPhim();
    XuatDSSinhVien(listStudent);
    Console.ReadKey();
}
```


- Để giúp kiểm tra được nhanh và đa dạng nhiều dữ liệu (không mất thời gian nhập từ bàn phím) sinh viên nên tạo 1 hàm trả về List danh sách sinh viên

```
private static List<Student> LayListSinhvienTest() //hàm tạo list dữ liệu để test nhanh
{
    List<Student> listStudent = new List<Student>();
    listStudent.Add(new Student("1", "Nguyễn văn A", 10, "CNTT"));
    listStudent.Add(new Student("2", "Nguyễn văn B", 6, "NNA"));
    listStudent.Add(new Student("3", "Nguyễn văn C", 3, "CNTT"));
    return listStudent;
}
```

- Lúc đó ở hàm main() chúng ta có thể dễ dàng điều chỉnh gọi hàm nhập và thực hiện 1 số yêu cầu truy xuất

```
static void Main(string[] args)
{
    // List<Student> listStudent = LayListSinhvienTuBanPhim();
    List<Student> listStudent = LayListSinhvienTest();
    XuatDSSinhvien(listStudent);

    //1. Try xuất tìm các sinh viên thuộc khoa "CNTT"
    List<Student> listKQ = listStudent.Where(p => p.Faculty == "CNTT").ToList();
    if (listKQ.Count > 0)
    {
        Console.WriteLine("--Danh sách sinh viên thuộc khoa CNTT ");
        XuatDSSinhvien(listStudent);
    }
    else
    {
        Console.WriteLine("--Không tìm thấy sinh viên thuộc khoa CNTT");
    }
    Console.ReadKey();
}
```

Bài tập 2: Tạo thêm 1 project ở trong cùng solution Lab01 có tên là “**Lab01-02**”.

- ✓ Viết lại chương trình từ bài tập 1 trên theo cách tạo thêm một lớp là **Person** làm lớp cơ sở cho lớp **Student**. Chọn Mã số, Họ tên làm field để đưa lên lớp **Person**.
- ✓ Thêm thông tin lớp giảng viên **Teacher** kế thừa từ lớp **Person**. Mỗi giảng viên đều có Mã số, Họ Tên và Địa chỉ.
- ✓ Ở hàm **Main()** cho phép nhập vào tổng số N đối tượng được đưa vào **List** danh sách **Person**, ở mỗi lần nhập liệu user có quyền chọn là nhập thông tin cho **Student** hoặc **Teacher**. Xuất danh sách thông tin vừa nhập liệu trên.
- ✓ Thực hiện một số yêu cầu sau trên List danh sách nhập liệu
 - 2.1 Tìm kiếm danh sách các Sinh Viên có điểm trung bình cao nhất thuộc khoa “**CNTT**”.
 - 2.2 Xuất ra danh sách tất cả sinh viên có điểm trung bình <5 và thuộc khoa “**CNTT**”
 - 2.3 Viết phương thức tìm kiếm dữ liệu. Nhập vào một “Tên” để tìm kiếm, Xuất ra tất cả

danhsach những sinh viên hoặc giảng viên mà có tên chứa tên tìm kiếm không phân biệt các kí tự hoa thường.

```
// Tìm kiếm danh sách các Sinh Viên có điểm trung bình cao nhất thuộc khoa "CNTT" (Trong
Trường hợp Student kế thừa từ Person và không có điểm. Sinh viên Sử dụng từ khóa is, as
trong ngôn ngữ C#

//b1: lay ra list person ma la sinh vien khoa CNTT
var listTempStudent = listPerson.Where(p => (p is Student) && (p as Student).Faculty ==
"CNTT").ToList();

//b2: lay diem cao nhất trong list tìm kiếm
var maxScore = listTempStudent.Max(p => (p as Student).AverageScore);

//b3: lấy danh sách sinh viên
var ListMaxStudent = listTempStudent.Where(p => (p as Student).AverageScore ==
maxScore);
```

Bài tập 3: Thêm 1 project ở trong cùng solution Lab01 có tên là **“Lab01-03”**.

Xây dựng chương trình quản lý danh sách hoá đơn tiền điện của khách hàng. Thông tin bao gồm các loại khách hàng:

- Khách hàng Việt Nam: mã khách hàng, họ tên, ngày ra hoá đơn (ngày, tháng, năm), đối tượng khách hàng (sinh hoạt, kinh doanh, sản xuất): số lượng (số KW tiêu thụ), đơn giá, định mức.
Thành tiền được tính như sau:
 - Nếu số lượng <= định mức thì: thành tiền = số lượng * đơn giá.
 - Ngược lại thì: thành tiền = số lượng * đơn giá * định mức + số lượng KW vượt định mức * Đơn giá * 2.5.
- Khách hàng nước ngoài: mã khách hàng, họ tên, ngày ra hoá đơn (ngày, tháng, năm), quốc tịch, số lượng, đơn giá. Thành tiền được tính = số lượng * đơn giá.

Thực hiện các yêu cầu sau:

- ✓ Xây dựng các lớp với chức năng thừa kế
- ✓ Nhập xuất danh sách các hóa đơn khách hàng.
- ✓ Xuất ra các hoá đơn trong tháng 09 năm 2019 (của cả 2 loại khách hàng).
- ✓ Tính tổng số KW tiêu thụ vượt định mức của khách hàng Việt Nam.
- ✓ Nhập vào một mã khách hàng để tìm kiếm. Xuất ra tổng thành tiền của mã số khách hàng vừa tìm kiếm (nếu có).

-----Hết Lab 01-----