Дәріс 1. Жүйелік бағдарламалаудың негізгі ұғымдары.

Кіріспе

«Жүйелік бағдарламалау» деген ұғым компьютерлердің операциялық жүйелерімен (ОЖ) байланысты. Тарихи, ең бірінші үлкен программалар болып операциялық жүйелер саналады.

Қолданбалы программалар оперативті жадының бос көлемімен шектелген, олар көбінесе есептеумен байланысты есептерді шешкен.

Қазырғы заманда қолданбалы немесе жүйелік программаларының күрделігі зорырақ екенін ажырату өте күрделі, себебі кей бір қолданбалы программалардың өздері үлкен жүйелік жобалар және олар күрделіктері бойынша ОЖ кем емес.

«Жүйелік программалау» пәнінде алгоритмдер және ОЖ қатысты есептерді шешу сұрактар қарастырылады.

1.1 Операциялық жүйелер ұғымы және олардың міндеті

Компьютер ұғымына физикалық, аппарттық, логикалық және ақпараттық ресурстар кіреді.

Физикалық ресурстар ол компьютердің құрамына кіретін құрылымдар – процессор, енгізу-шығару құрылғысы, жады және т.б. К физическим ресурсам относятся различные устройства компьютера – процессор, устройства ввода-вывода, память и т.д.

Логикалық ресурстарға деректер, программалар жатады.

Компьютердің бәр ресурстары жүйелік ресурстар деп аталады.

Соған байланысты операциялық жүйені компьютердің жүйелік ресурстарына қатынау және оларды басқару программалар комплесі деп анықтауға болады. Тогда операционную систему можно определить как комплекс программ, которые обеспечивают доступ и управление системными ресурсами компьютера.

Компьютерде ОЖ басқаруымен жумыс істейтін программалар қолданушылар программалары деп аталады. Программы, работающие на компьютере под управлением ОС, называют пользовательскими программами.

Бір есепті шешуге арналған қолданушылар программалары приложение деп аталады.

Егер ОЖ тек бір колданушылар программасын орындауға мүмкіндік берсе онда ол ОЖ бір қолданушы немесе бір программалық деп аталады. Если ОС позволяет выполнять только одну пользовательскую программу, то ОС называется однопользовательской или однопрограммной.

Егер ОЖ бірнеше колданушылар программасын орындауға мүмкіндік берсе онда ол ОЖ көп қолданушы немесе мультипрограммалық деп аталады. Если ОС позволяет одновременно выполнять несколько пользовательских программ, то ее называют многопользовательской или мультипрограммной.

Егер ОЖ тек бір процессорлы компьютерде жұмыс істей алса онда ол бірпроцессорлы ОЖ деп аталады.

Если ОС может работать только на компьютере с одним процессором, то ее называют однопроцессорной.

Егер ОЖ бірнеше процессорлы компьютерде жұмыс істей алса онда ол мультпроцессорлы ОЖ деп аталады.

Если ОС может работать на компьютере, аппаратный ресурс которого может содержать один или несколько процессоров, то ОС называется мультипроцессорной.

Қолданушыға катынассыз нақты уақыт тәртіптемесімен жұмыс жасайтын ОЖ нақты уақыт тәртіптеме ОЖ деп аталады. Ондай ОЖ кейбір технологиялық процесстерді басқаруға арналған.

Операционные системы, работающие без участия пользователя в режиме реального времени, называют ОС реального времени. Такие операционные системы предназначены для управления некоторых технологических процессов.

1.2 Қолданбалы программалаудың Win32 API интерфейсінің ұғымы және міндеті.

Әрбір ОЖ қолданушыға жүйелік ресурстарды қолдануға мүмкіндік береді - қолданушы интерфейсі. Қолданушы интерфейсіне ОЖ басқаруымен жумыс жасайтын қосымшаларды программалауға қажет типтер, тұрақтылар, айнымалылар, функциялар және класстар жиындары кіреді.

Каждая ОС предоставляет пользователю возможность использования практически всех своих системных ресурсов – пользовательский интерфейс, включающий наборы типов, констант, переменных, функций и классов для программирования приложений, работающих под управлением ОС.

Осы кезде қолданылатын Windows ОЖ-дің интерфейстері бірдей - Win32 API (Application Programming Interface).

Все, работающие в настоящий момент ОС, имеют практически одинаковый интерфейс программирования приложений – Win32 API (Application Programming Interface).

Win32 API интерфейсі, тек қана қолданбалы программалауда ғана емес, жүйелік программалауда да қолданылады.

Естественно Win32 API используется не только в прикладном, но и в системном программировании.

Win32 API функцияларын келесі катигорияларға бөлуге болады:

- базалық сервистер (Base Services);
- ортақ басқару элементтерінің кітапханасы (Common Control Library);
- графикалық құрылғылардың интерфейсі (Graphics Devices Interface);
- желілік сервистер (Network Services);
- қолданушы интерфейсі (Uses Interface);
- Windows қатынауды басқару (Windows Access Control);
- Windows оболочкасы (қабықшасы) (Windows Shell);
- Windows жүйесі туралы ақпарат (Windows System Information).

Условно все функции Win32 API можно подразделить на следующие категории:

- базовые сервисы (Base Services);
- библиотека общих элементов управления (Common Control Library);
- интерфейс графических устройств (Graphics Devices Interface);
- сетевые сервисы (Network Services);
- интерфейс пользователя (Uses Interface);
- управление доступом для Windows (Windows Access Control);
- оболочка Windows (Windows Shell);
- информация о системе Windows (Windows System Information).

Қолданбалы бағдарламаларды жазумен байланысты пәндерде, негізінен қолданушы интерфейсінің функциялары оқытылады.

Желілік қызметтің функциялары жергілікті желілердің жұмысымен байланысты пәндерде оқытылады.

Графикалық құрылғылар интерфейсі түрлі ойын бағдарламаларын жазған уақытта қолданылады.

Win32 API басқа функцияларының тағайындалуларымен қолданыстары негізінен «Жүйелік бағдарламалау» пәніде оқытылады. .

1.3 Понятие объектов и дескрипторов в Windows объектілерімен дескрипторлары түсініктері

Win32 API функцияларын қолдану объетілерді қолдануға негізделген, әрі бұл уақытта объектіні класс айнымалы деген объектінің «классикалық» анықтамасы Win32 API функцияларын әзірлегеннен соң қолданылғанын есте сақтау керек.

Windows-та объект деп жүйелік ресурсы бар мәліметтер құрылымы аталады. Іс жүзінде бұл объект Windows бөлінген және объект туралы ақпарат сақталған жады облысы. Объектілер арнайы функциялардың көмегімен құрылады және оларға қатынау тек ОЖ функцияларының көмегімен ғана мүмкін. Win32 API үш санаттың объектілерін құра алады:

– қолданушы интерфейсінің объектісі (User);

–графикалық құрылғылар интерфейсінің объектісі (Graphics Devise Interface);

– ОЖ ядросының объектілері (Kernel).

Мы будем рассматривать только объекты ядра операционной системы.

При создании объекта ему присваивается имя (идентификатор), которое называется дескриптором (handle). В ОС дескриптор объекта представляет собой запись в таблице дескрипторов, которая содержит адрес объекта и средства для идентификации типа объекта.

В Win32 API дескрипторы имеют тип HANDLE.

При обращении к объекту необходимо указывать его дескриптор.

По окончании работ с объектом его необходимо закрывать.

1.4 Динамикалық қосылатын библиотекалар түсінігі Понятие динамически подключаемых библиотек

«Динамикалық қосылатын библиотекалар (dynamic-link libraries, DLL)– Windows ОЖ ең алғашқы нұсқасынан бастап оның іргетасы. DLL-да Win32 API функцияларының барлығы бар. Ең мағызды үш DLL: Kernel32.dll (жадыны, үдерістерді және ағындарды басқару), User32.dll (қолданушы интерфейсін құптау, оның ішінде терезелерді құру және хабарламаларды жіберу функцияларымен байланыстыларын да) және GDI32.dll (графика және мәтінді шығару).

Windows-та функциялары мамандандырылған тапсырмаларды орындауға арналған басқа да DLL бар. Мысалы, AdvAPI32.dll, мұнда реестрмен жұмыс және оқиғаларды тіркеу объектілерін қорғауға арналған функциялар бар, ал ComCtl32.dll стандартты басқару элементтерін құптайды.

Динамикалық қосылатын библиотекалар не үшін керек? Міне, DLL қолдану керектігінің кейбір себептері ғана:

– қосымшаның қымзететуін кеңейту. DLL-ді үдерістің адрестік кеңістігіне динамикалық түрде жүктеуге болады, ал бұл қосымшаға одан қандай әркеттер қажет екенін анықтап, керекті кодты жүктеуге мүмкіндік береді. Сондықтан, бір компания қандай да бір қосымшаны құрып, оның қызмет етуін басқа компаниялардың DLL есебінен кеңейте алады;

– түрлі бағдарламалау тілдерін қолдану мүмкіндігі. Сізде қосымшаның қандай да бір бөлігін қай тілде жазу таңдауы бар;

– жобаны қарапайым басқару. Егер бағдарламалық өнімді әзірлеу барысында оның жекелеген модульдерін әртүрлі топтар құрса, онда DLL қолданған уақытта бұл сияқты жобаны басқару әлдеқайда жеңіл;

– жадыны үнемдеу. Егер бір DLL бірнеше қосымша пайдаланса, онда оперативті жадыда оның осы қосымшалардың барлығына қолжетімді болатын бір ғана данасы сақталынады.

– ресурстарды бөлу. DLL-да диалогтық терезенің, жолдың шаблондары, белгілер және биттік карталар (растрлық суреттер) сияқты ресурстар болуы мүмкін. Бұл ресурстар кез келген бағдарлама үшін қолжетімді болады;

– түрлі платформалардың ерекшеліктерімен байланысты мәселелерді шешу. Windows-тың әртүрлі нұсқаларында әртүрлі функциялар жиыны бар. Көбіне әзірлеушілерге олар өздері қолданып отырған нұсқасында бар жаңа функциялар керек. Егер бұл функциялар сіздің Windows нұсқаңызда жұмыс істемейтін болса, онда Сіз бұл сияқты қосымшаны іске қоса алмайсыз: жүктеуші оны іске қосудан бас тартады. Бірақ, егер бұл функциялар жеке библиотекада (DLL) болатын болса, онда Сіз бағдарламаны тіптен Windows ертеректегі нұсқасында да жүргізесіз (Джеффри Рихтер Мамандарға арналған Windows, 476-477 б.).

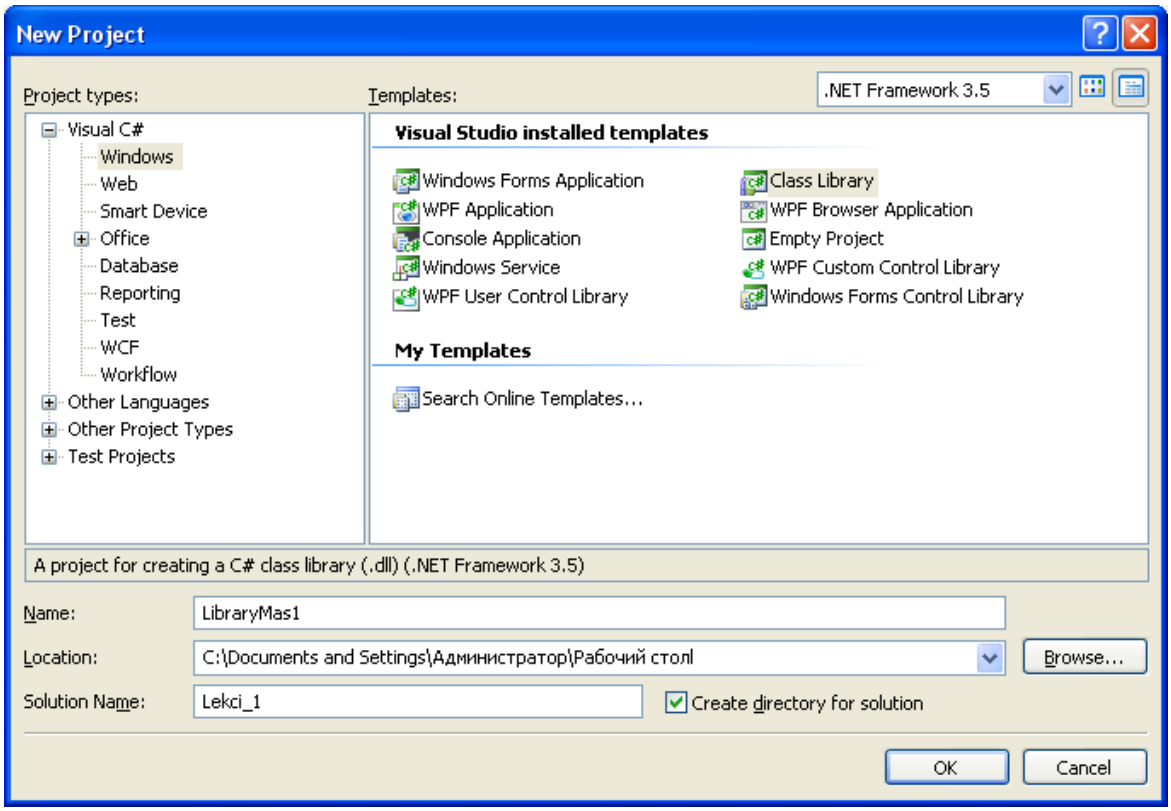
1.5 Кейбір тапсырмалар үшін библиотеканы әзірлеу

Бір өлшемді массивта бүтін сандарды өңдеумен байланысты есепке арнап DLL оқу библиотекасын әзірлейік. Бұл тек үйрету тапсырмасы, мұнда DLL библиотекасын құру технологиясы қарастырылады. Әрі DLL библиотекаларды C# тілінің консольді қосымшасында қолдану сұрақтары қарастырылған.

Біздің DLL библиотекамызда бүтін сандар массивімен жұмыс істейтін тек үш әдіс қана болады:

- максималды мәнді іздеу;
- кему реті бойынша сұрыптау;
- массивтің барлық элементтерінің мәндерінің қосындысын есептеу.

Visual Studio 2008 іске қосып, жобаны құруға көшеміз. Жоба типі ретінде "Class Library" типін көрсетіміз. DLL библиотекасын құру терезесінің өрістерін 1.1-суретіне сәйкес қайта анықтап аламыз.



1.1-сурет – DLL библиотеканы құру

Name өрісіне құрылатын DLL аты беріледі. Құрылатын DLL аты LibraryMas1 болсын.

Location өрісіне ішінде жоба болатын Шешім сақталынатын директория көрсетілед. Компьютердің жұмыс столын (рабочий стол) көрсетеміз.

Solition Name терезесінде «Шешім» аты беріледі. Бірінші дәрістің барлық жобалары бір «Шешімге» біріктірілгенін көрсететін Lekci_1 атауын қолданамыз.

Solition Name өрісінде «Шешімді» орналастыру үшін жаңа директория құру керектігін көрсететін "Create directory for solution" элементі таңдалған.

Жоба типін растағаннан соң, ОК бастырмасын басу арқылы келесі кодты аламыз:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LibraryMas1
{
    public class Class1
    {
    }
}
```

"Class1" атын "MyMas" атына өзгертейік, ол үшін жоба кодының терезесінде өзгертілетін идентификаторды белгілеп алып, одан соң басты мәзірден Refactor пунктін және Rename ішкі пунктін таңдаймыз. Ашылған терезеде жаңа атты көрсетеміз. Сонда идентификатордың атын өзгертуді талап ететін жерлердің барлығы көрсетіледі. Біздің жағдайымызда тек бір ғана ауыстыру болады, бірақ жалпы алғанда ауыстырулар көп болуы мүмкін, сондықтан барлық кірулерді автоматты түрде ауытыру өте пайдалы болады.

Құрылып жатқан DLL библиотеканы тапсырманың шартына сәйкес үш әдістің кодымен толықтырайық:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LibraryMas1
{
    public class MyMas
    {
        public static int maxMas(ref int[] masi, int n)
        {
            int max = masi[0];
            for (int i = 1; i < n; i++)
                if (max < masi[i]) max = masi[i];
            return max;
        }
        public static void sortMas(ref int[] masi, int n)
        {
         int b = 0;
         for (int i = 0; i < n - 1; i++)
          for (int j = i + 1; j < n; j++)
           if (masi[i] < masi[j])
            { b = masi[i]; masi[i] = masi[j]; masi[j] = b; }
        }
        public static int symMas(ref int[] masi, int n)
        {
```

```
            int sym = 0;
            for (int i = 0; i < n; i++)
                sym=sym+masi[i];
            return sym;
        }
    }
}
```

DLL библиотеаны құрудың ақырғы кезеңінде екі әрекетті орындау керек.

Біріншіден, «Class1.cs» файлының атын «MyMas.cs» атына ауыстыру керек. Өйткені, класты сақтап тұрған файл атымен кластың аты бүрдей болу керек. Файл атын ауыстыру Solition Explorer жоба терезесінде іске асады. Файл атын ерекшелеп алғаннан соң, тінтуірдің оң жақ батырмасын басып, Rename командасын таңдап аламыз. Жаңа атауды енгіземіз – өзгерістер жобаның барлық элементтерінде болады.

Екіншіден, құрылған DLL библиотеканы компиляция жасау керек, ол үшін Басты мәзірден Build -> Build Solition пунктін таңдаймыз. Сәтті компиляция нәтижесінде dll кеңейтілімі бар файл құрылады.

1.6 Құрылған DLL библиотекиану қолдану

Құрылған DLL библиотеканы қолданатын массивпен жұмс істеуге арналған жай ғана үйрену бағдарламасын қарастырайық. Бағдарлама келесі тапсырманы шешуге арналған.

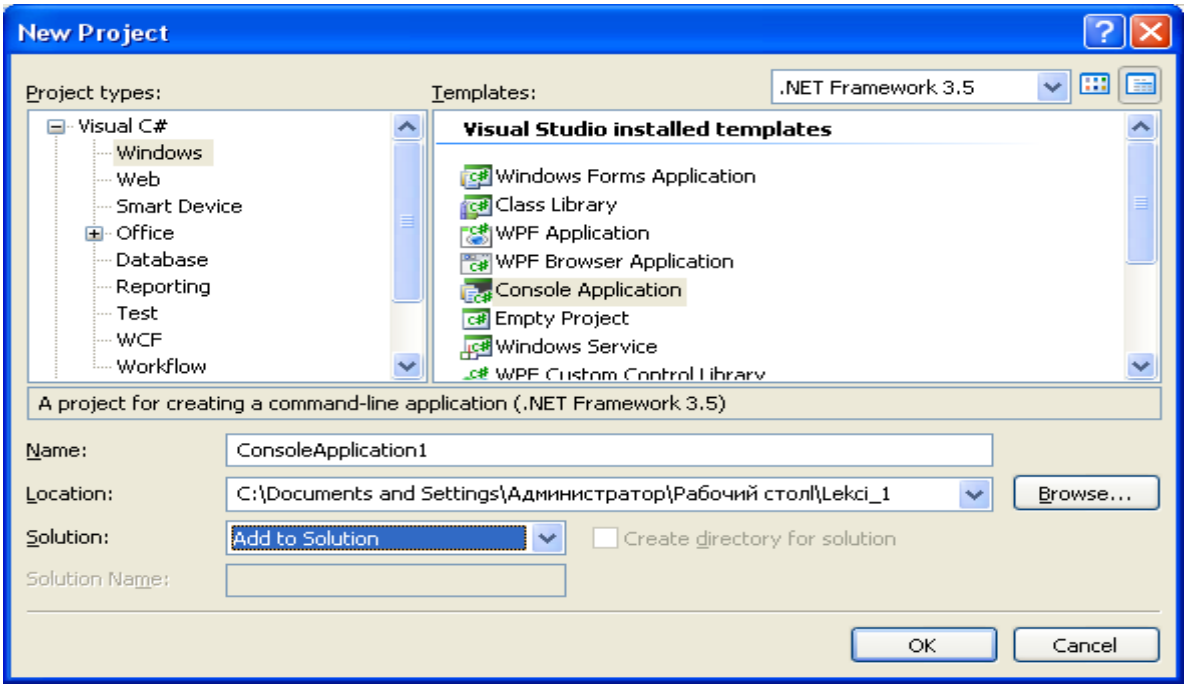
Тапсырма 1.1. минус 50-ден 50-ге дейінгі диапазондағы 20 кездейсоқ бүтін сандардан тұратын массив тұрғызу. Массивтің барлық элементтерінің қосындысын тауып, оны басып шығару. Массив мәндерін 1 разрядқа солға қарай жылжыту. Массив элементтерін кему реті бойынша сұрыптауды орындау. Бағдарлама мәзір болу керек.

DLL библиотеканы қолдану нұсқаларының бірі DLL «Шешіммен» бірге қосымшаны құру. Ол үшін бұрын DLL библиотекада құрылған жобаны қарапайым жолмен ашып, және онда консольді қосымша жобасын 1.2-суретке сәйкес құрамыз.

Name өрісінде бастапқыда беріліп тұрған мәнді қалтырамыз.

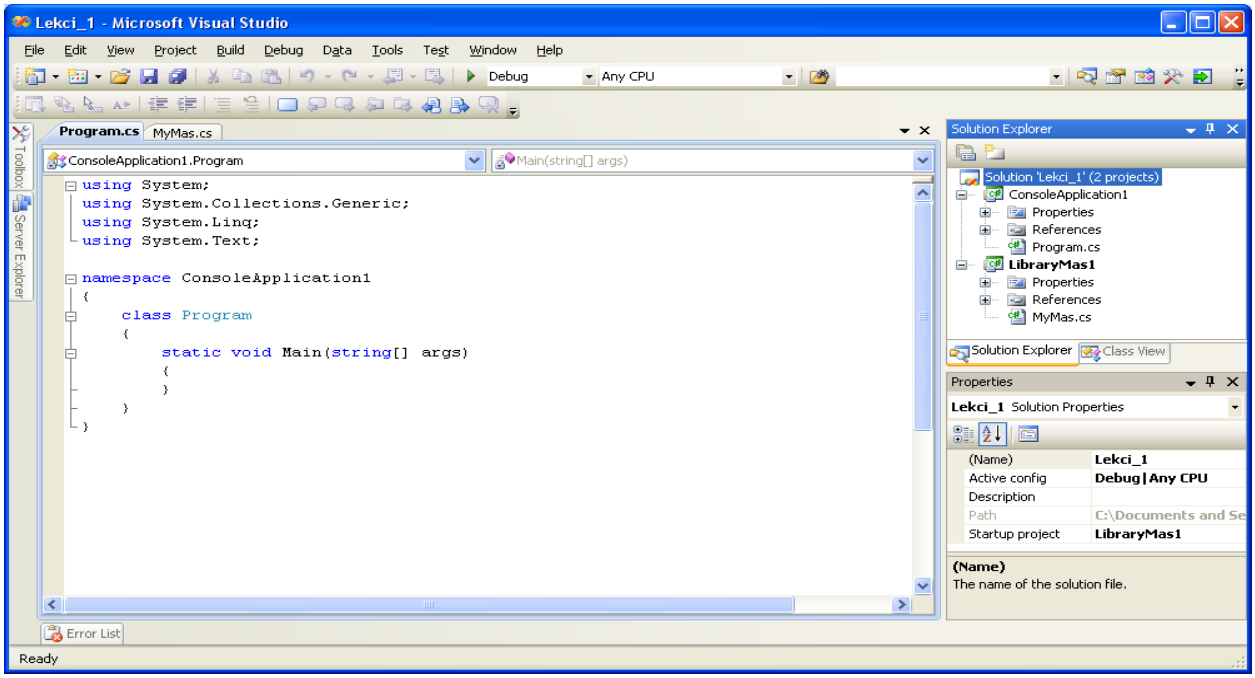
Location өрісінде бастапқыда беріліп тұрған мәнді қалтырамыз.

Solition өрісінде құрылатын қосымша бар «Шешімге» қосылатын "Add to Solution" мәнін таңдап аламыз.



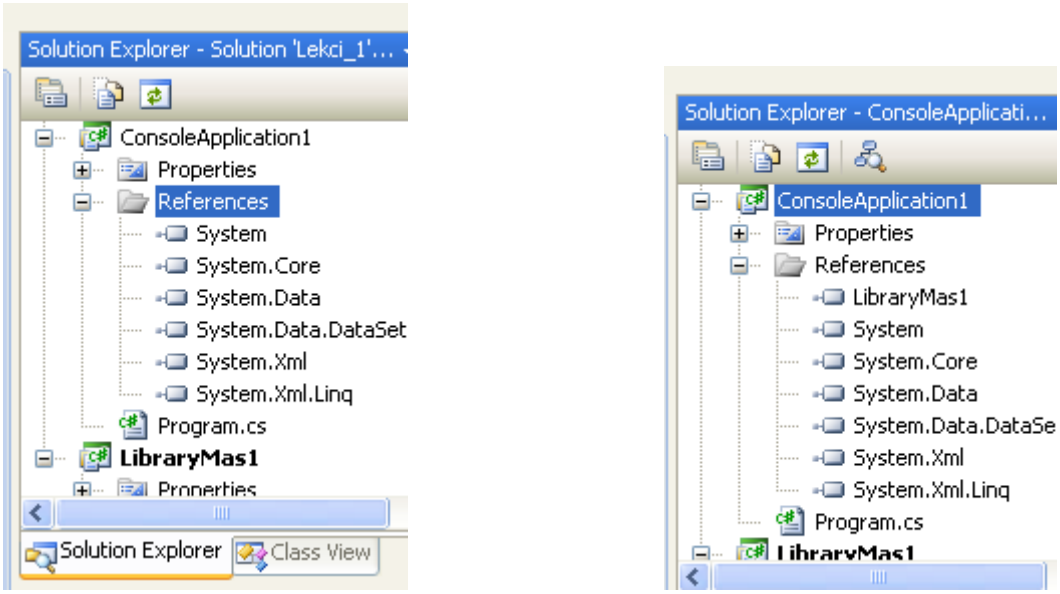
1.2-сурет – Консольді қосымшаны құру

Консольді қосымшаның кодының даярламасы 1.3-суретте көрсетлген.



1.3-сурет – Құрылған консольді қосышаның бастапқы терезесі

«Solition Explorer» терезесінде екі жоба бар екендігіне назар аударамыз. Еі жобада бір шешімде тұр, бірақ олар бір-бірімен байланысты емес. Оны тексеру оңай, «Solition Explorer» терезесінде «References» каталогын – 1.4-а –суретті қара.



a b

1.4-сурет – Консольді қосымшалардың байланысы

Байланыстың болмауы DLL библиотеканың әдістерін біздің консольді жобамызда қолдануды мүмкін етпейді.

Консольді қосымшамен DLL библиотеканың байланысын (DLL LibraryMas1 бар жобаға сілтеме қосу) ұйымдастыру үшін «Solition Explorer» терезесінде консольді қосымшаның атын (ConsoleApplication1) таңдап алып, тінтуірдің оң жақ батырмасына басқанда ашылатын контекстік мәзірден "Add Reference" пунктін таңдап аламыз. Ашылған сілтемелерді қосу терезесінен "Projects" бетін таңдаймыз. LibraryMas1 жобасы «Шешімге» қосылған болғандықтан, ол автоматты түрде ашылған терезеде пайда болады. Байланысты орнатуды ОК батырмасына басу арқылы растаймыз. DLL LibraryMas1-ге сілтеме консольді қосымшаның "References" каталогында пайда болады. 1.4 – b –суретінде DLL LibraryMas1-ге сілтеме сілтемелер тізімінің жоғары жағында пайда болғаны көрінеді. Енді жобалар өзара байланысқан, DLL ұсынылатын әдістер консольді қосымшадан қол жетімді болады.

DLL LibraryMas1-ге сілтемені Project-> Add Reference режимін қолдана отырып та орнатуға болады.

Егер сілтемені «Шешімге» кірмеген жобаға орнату керек болса, онда сілтемелерді қосу терезесінде жобаға апаратын жолды көрсету керек.

Тапсырманың шарттарына сай бағдарлама кодын әзірлейміз.

Бағдарламаның коды:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
```

```
 class Program
 {
  public static void sozd(ref int[] ma)
  {
   Random rnd = new Random();
   for (int i = 0; i < 20; i++)
    ma[i] = rnd.Next() % 101 - 50;
   Console.WriteLine("Массив создан !!");
  }
  public static void zadvig(ref int[] ma)
  {
   int k;
   for (int i = 0; i < 19; i++)
   {
    k = ma[i]; ma[i] = ma[i + 1]; ma[i + 1] = k;
   }
   Console.WriteLine("Сдвиг массива на 1 разряд выполнен !");
  }
  public static void prinmas(int[] ma)
  {
   for (int i = 0; i < 20; i++)
    Console.Write(" {0}", ma[i]);
   Console.WriteLine();
  }
  static void Main()
  {
   int[] a = new int[20];
   int k = 0;
   int s = 0,n = 20;
   string buf;
   while (k < 6)
   {
    Console.WriteLine("1 - Создать массив 20 чисел");
    Console.WriteLine("2 - Напечатать массив");
    Console.WriteLine("3 - Найти и напечатать сумму всех
элементов массива");
    Console.WriteLine("4 - Выполнить сдвиг значений массива на 1
разряд влево");
    Console.WriteLine("5 - Выполнить сортировку элементов
массива в порядке убывания");
    Console.WriteLine("6 - Выход из программы");
    Console.WriteLine("Введите пункт меню программы");
    buf = Console.ReadLine();
    k = Convert.ToInt32(buf);
    switch (k)
    {
     case 1: sozd(ref a); break;
     case 2: prinmas(a); break;
     case 3:
      {
       s = LibraryMas1.MyMas.symMas(ref a, n);
       Console.WriteLine("Сумма элементов массива = {0}",s);
      }; break;
```

```
            case 4: zadvig(ref a); break;
            case 5: LibraryMas1.MyMas.sortMas(ref a,n); break;
            default: break;
          }
        }
      }
    }
  }
```

Егер жобаны орындауға жіберуге талпынсақ, онда іске қосылатын жоба анықталмаған деген хабарлама шығады. Бұл сияқты жағдай «Шешімде» бірнеше жоба болғанда орын алады. Бірінші қосылатын жобаны анықтау үшін осы жобаның редакторы терезесінде тұрып, келесі Project -> Set as StartUp Project командасын беру керек.

Бағдарлама жұмысы:

1 - Создать массив 20 чисел
2 - Напечатать массив
3 - Найти и напечатать сумму всех элементов массива
4 - Выполнить сдвиг значений массива на 1 разряд влево
5 - Выполнить сортировку элементов массива в порядке убывания
6 - Выход из программы
Введите пункт меню программы
1
Массив создан !!
1 - Создать массив 20 чисел
2 - Напечатать массив
3 - Найти и напечатать сумму всех элементов массива
4 - Выполнить сдвиг значений массива на 1 разряд влево
5 - Выполнить сортировку элементов массива в порядке убывания
6 - Выход из программы
Введите пункт меню программы
2
-36 -2 -3 35 -10 28 -32 11 -21 -32 -35 13 6 3 -45 32 -27 -5 -36 15
1 - Создать массив 20 чисел
2 - Напечатать массив
3 - Найти и напечатать сумму всех элементов массива
4 - Выполнить сдвиг значений массива на 1 разряд влево
5 - Выполнить сортировку элементов массива в порядке убывания
6 - Выход из программы
Введите пункт меню программы
3
Сумма элементов массива = -141
1 - Создать массив 20 чисел
2 - Напечатать массив
3 - Найти и напечатать сумму всех элементов массива

4 - Выполнить сдвиг значений массива на 1 разряд влево
5 - Выполнить сортировку элементов массива в порядке убывания
6 - Выход из программы
Введите пункт меню программы
4
Сдвиг массива на 1 разряд выполнен !
1 - Создать массив 20 чисел
2 - Напечатать массив
3 - Найти и напечатать сумму всех элементов массива
4 - Выполнить сдвиг значений массива на 1 разряд влево
5 - Выполнить сортировку элементов массива в порядке убывания
6 - Выход из программы
Введите пункт меню программы
2
-2 -3 35 -10 28 -32 11 -21 -32 -35 13 6 3 -45 32 -27 -5 -36 15 -36
1 - Создать массив 20 чисел
2 - Напечатать массив
3 - Найти и напечатать сумму всех элементов массива
4 - Выполнить сдвиг значений массива на 1 разряд влево
5 - Выполнить сортировку элементов массива в порядке убывания
6 - Выход из программы
Введите пункт меню программы
5
1 - Создать массив 20 чисел
2 - Напечатать массив
3 - Найти и напечатать сумму всех элементов массива
4 - Выполнить сдвиг значений массива на 1 разряд влево
5 - Выполнить сортировку элементов массива в порядке убывания
6 - Выход из программы
Введите пункт меню программы
2
35 32 28 15 13 11 6 3 -2 -3 -5 -10 -21 -27 -32 -32 -35 -36 -36 -45
1 - Создать массив 20 чисел
2 - Напечатать массив
3 - Найти и напечатать сумму всех элементов массива
4 - Выполнить сдвиг значений массива на 1 разряд влево
5 - Выполнить сортировку элементов массива в порядке убывания
6 - Выход из программы
Введите пункт меню программы