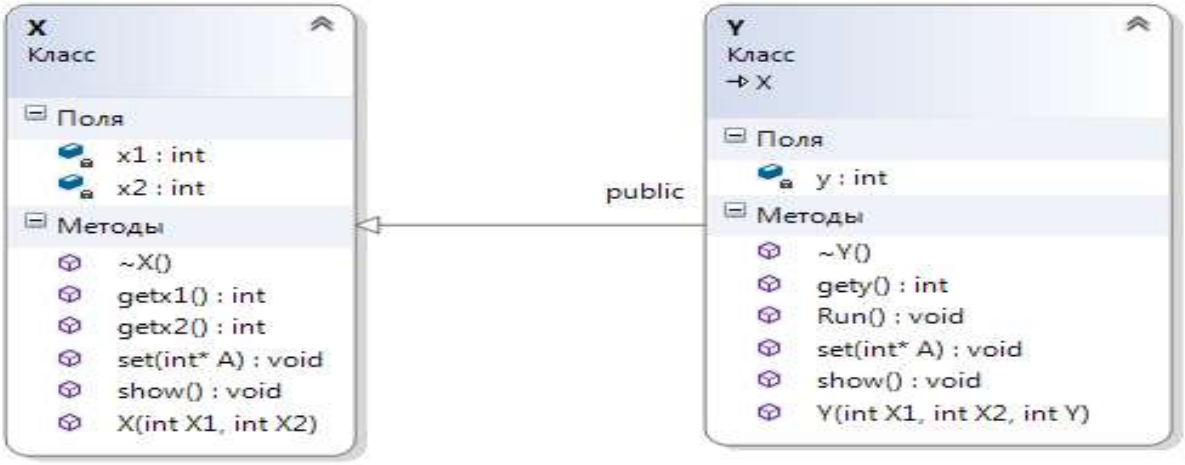
Общее задание

По диаграмме классов создать иерархию классов и продемонстрировать их работу. Программа должна создать базовый и производный классы согласно диаграмме классов:



Варианты заданий

Задание А

Продемонстрировать работу классов, используя ***ОБЪЕКТ производного*** класса
- создать объект производного класса,
- вызвать метод просмотра базового класса,
- вызвать метод просмотра производного класса,
- вызвать метод Run. (таблица 4.1)
- переустановить значения, вызвать методы просмотров и Run.
- создать аналогичную диаграмму классов

Задание Б

Продемонстрировать работу классов, используя ***УКАЗАТЕЛЬ на ПРОИЗВОДНЫЙ*** класс:
- создать указатель на производный класс, связать с объектом производного класса,
- вызвать метод просмотра базового класса,
- метод просмотра производного класса,
- метод переустановки значений и вызов методов просмотра.

Задание В

Продемонстрировать работу классов, используя ***УКАЗАТЕЛЬ на БАЗОВЫЙ*** класс:
- создать указатель на базовый класс, связать с объектом производного класса,
- вызвать метод просмотра базового класса,
- метод просмотра производного класса,
- метод переустановки значений и вызов методов просмотра.

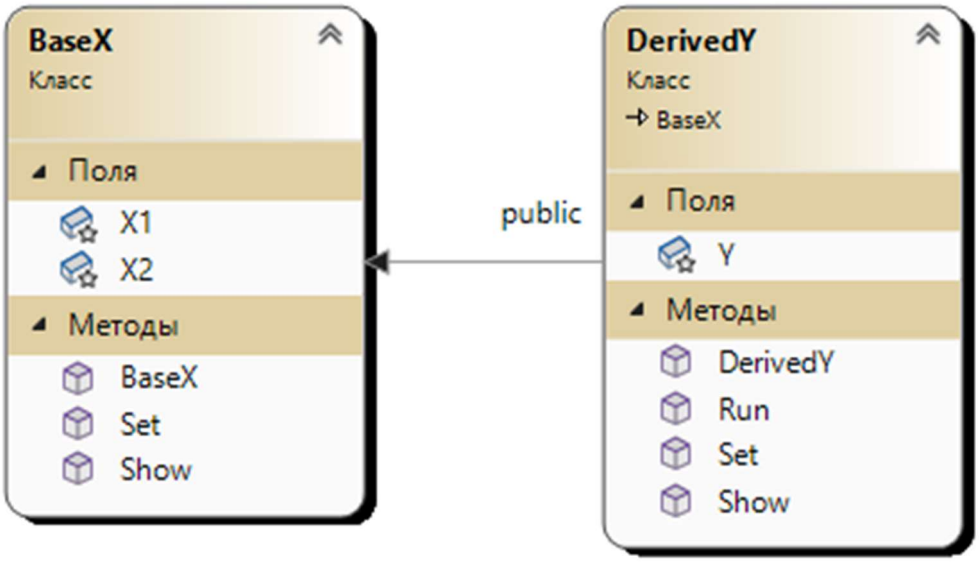
Задание Г

Объявить в базовом класса метод просмотра (печати) текущего состояния и метод переустановки объектов в новое состояние как *ВИРТУАЛЬНЫЕ* методы.
Продемонстрировать работу классов, используя ***УКАЗАТЕЛЬ на БАЗОВЫЙ*** класс:
- создать указатель на базовый класс, связать с объектом производного класса,
- вызвать метод просмотра базового класса,
- метод просмотра производного класса,
- метод переустановки значений и вызов методов просмотра.

Комментарий

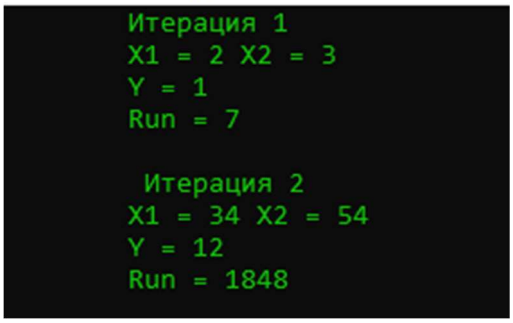
В программе совершенно не используется метод Get() данный на вашей диаграмме, поэтому я исключил его из своего кода. Также от варианта к варианту классы не меняются, поэтому я приведу их в ответе на первое задание, а далее буду записывать только реализацию в main(), за исключением последнего варианта.

Моя диаграмма классов



Мой код – Задание А

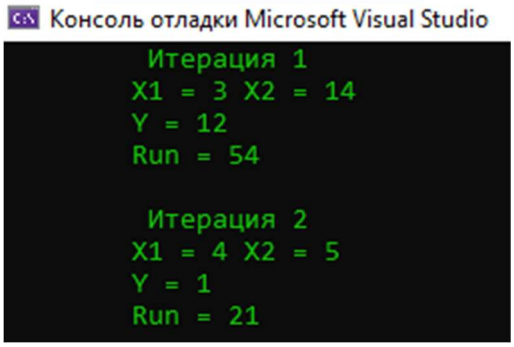
Консоль отладки Microsoft Visual Studio




```cpp
#include <iostream>
using namespace std;

class BaseX
{
protected:
	int X1;
	int X2;

public:
	BaseX(int x1, int x2) {
		X1 = x1;
		X2 = x2;
	}

	void Set(int* A) {
		X1 = A[0];
		X2 = A[1];
	}

	void Show() {
		cout << "X1 = " << X1 << " X2 = " << X2 << "\n";
	}
};

class DerivedY : public BaseX
{
protected:
	int Y;

public:
	DerivedY(int y = 0, int x1 = 0, int x2 = 0) : BaseX(x1, x2) {
		Y = y;
	}

	void Set(int* A) {
		Y = *A;
	}

	void Show() {
		cout << "Y = " << Y << "\n";
	}

	void Run() {
		cout << "Run = " << X1 * X2 + Y << "\n";
	}
};

int main()
{
	system("color A");
```

```
	setlocale(LC_ALL, "Rus");

	DerivedY newSon(1, 2, 3); // Создаем объект производного класса
	BaseX* pointer = &newSon; // Создаем ссылку типа базового класса на
производный класс
	/* Происходит связываение, так как срабатывает приведение типов ссылок,
	 * теперь оба класса "знают друг друга" и с помощью ссылки мы будем вывзвать
методы
	 * базового класса. К методам производного класса можно обращаться через
точку.
	 */

	cout << "Итерация 1\n";

	pointer->Show(); // Вызываем метод базового класса используя ссылку
	newSon.Show(); // Вызываем метод производного класса через оператор
	newSon.Run();

	cout << "\nИтерация 2\n";

	int newX[] = { 34, 54 };
	pointer->Set(newX); // Изменяем значения базового класса

	int newY = 12; int* A = &newY;
	newSon.Set(A); // Изменяем значения производного класса

	pointer->Show();
	newSon.Show();
	newSon.Run();
}
```

Мой код – Задание Б




```
int main()
{
	system("color A");
	setlocale(LC_ALL, "Rus");

	DerivedY NewClass(12, 3, 14); // Создаем объект производного класса
	DerivedY* pointer = &NewClass; //Указатель на производный класс типа производного класса

	/* В данном примере мы используем ссылку на наследуемый класс,
	 * благодоря этому мы можем использовать методы наследуемого класса по прямой ссылке,
	 * метобы базового класса мы можем вызывать используя приведение типов непосредственно
	 * к базовуму типу прямо перед вызовом функции.
	 */

	cout << "Итерация 1\n";

	((BaseX*)(pointer))->Show(); // Приводим ссылку и вызываем базовый класс
	pointer->Show(); // Не троегаем ссылку и вызываем класс наследника
	pointer->Run();

	cout << "\nИтерация 2\n";

	int newX[] = { 4, 5 };
	((BaseX*)(pointer))->Set(newX); // Изменяем значения базового класса

	int newY = 1; int* A = &newY;
	pointer->Set(A); // Изменяем значения производного класса

	((BaseX*)(pointer))->Show();
	pointer->Show();
	pointer->Run();
}
```

Мой код – Задание B

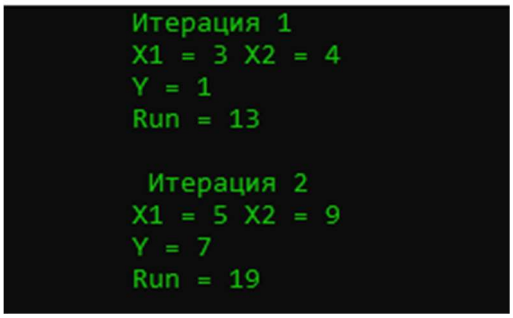
Консоль отладки Microsoft Visual Studio

```
Итерация 1
X1 = 33 X2 = 195
Y = 145
Run = 6580

 Итерация 2
X1 = 98 X2 = 12
Y = 73
Run = 1249
```

```
int main()
{
	system("color A");
	setlocale(LC_ALL, "Rus");

	DerivedY NewClass(145, 33, 195); // Создаем объект производного класса
	BaseX* pointer = &NewClass; // Создаем указатель на производный класс типа базового класса

	/* Происходит связываение, так как срабатывает приведение типов ссылок,
	 * теперь оба класса "знают друг друга" и с помощью ссылки мы будем вывзвать методы
	 * базового класса. К методам производного класса будем обращаться по ссылке,
	 * используя приведение типов
	 */

	cout << "    Итерация 1\n";

	pointer->Show(); // Вызываем метод базового класса используя ссылку
	((DerivedY*)(pointer))->Show(); // Приводим ссылку и вызываем производный класс
	((DerivedY*)(pointer))->Run();

	cout << "\n   Итерация 2\n";

	int newX[] = { 98, 12 };
	pointer->Set(newX); // Изменяем значения базового класса

	int newY = 73; int* A = &newY;
	((DerivedY*)(pointer))->Set(A); // Изменяем значения производного класса

	pointer->Show();
	((DerivedY*)(pointer))->Show();
	((DerivedY*)(pointer))->Run();
}
```

Мой код – Задание Г

Консоль отладки Microsoft Visual Studio




```
#include <iostream>
using namespace std;

class BaseX
{
protected:
	int X1;
	int X2;

public:
	BaseX(int x1, int x2) {
		X1 = x1;
		X2 = x2;
	}

	virtual void Set(int* A) {
		X1 = A[0];
		X2 = A[1];
	}

	virtual void Show() {
		cout << "	X1 = " << X1 << " X2 = " << X2 << "\n";
	}
};

class DerivedY : public BaseX
{
protected:
	int Y;

public:
	DerivedY(int y, int x1, int x2) : BaseX(x1, x2) {
		Y = y;
	}

	virtual void Set(int* A) override {
		Y = *A;
	}

	virtual void Show() override {
		cout << "	Y = " << Y << "\n";
	}

	void Run() {
		cout << "	Run = " << X1 * X2 + Y << "\n";
	}
};

int main()
{
	system("color A");
```

```
	setlocale(LC_ALL, "Rus");

	BaseX NewClassX (3, 4); // Создаем обект базового класса
	DerivedY NewClassY(1, 3, 4); // Создаем обект базового класса

	BaseX* pointerX = &NewClassX; // Создаем указатели по которым потом обратимся
к виртуальным функциям
	BaseX* pointerY = &NewClassY;

	cout << "    Итерация 1\n";
	pointerX->Show(); // Т.к указатель на Базовый клас => вызывается метод
Базового класса
	pointerY->Show(); // Т.к указатель на Производный клас => вызывается метод
Производного класса
	NewClassY.Run();  // Т.к функция не виртуальная, мы не можем вызвать ее через
указатель

	cout << "\n   Итерация 2\n";

	int newX[] = { 5, 9 };
	pointerX->Set(newX); // Изменяем значения базового класса

	int newY = 7; int* A = &newY;
	((DerivedY*)(pointerY))->Set(A); // Изменяем значения производного класса

	pointerX->Show();
	pointerY->Show();
	NewClassY.Run();
}
```