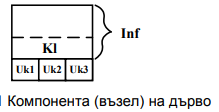
Доклад

По Алгоритми и структури от данни

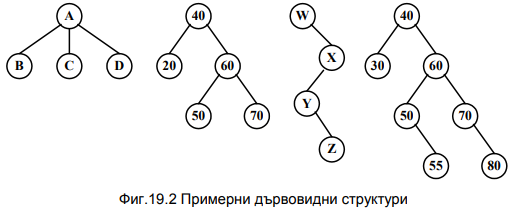
Тема – Дървовидни структури от данни

1. Дървовидните структури от данни
2. Увод в теорията на графите
3. Съпоставяне на дърво и граф
4. **Дървовидните структури от данни**

* Дървото е съвкупност от свързани компоненти, организирани в йерархична структура по своите ключови стойности.
* Компоненти (възли) на дървото. Компонентите на дървото са динамични
* информационно поле Inf - обикновено е от тип запис с полета, определени от конкретните цели, за които се създава дървото. Едно от полетата трябва да е уникално и то изпълнява ролята на ключ. Тук ще приемем, че информационното поле има само две полета: Ime и Kl, където Kl е ключ;
* указателни полета Uk1, Uk2,..., UkN, - указващи компонентите-наследници.Очевидно, за да е дърво динамичната структура, указателните полета трябва да са най-малко две.



* На диаграмите възлите се означават с кръгчета. В кръгчето обикновено се дава само стойността на ключа на възела. По-долу са показани няколко дървовидни структури.



**Основни операции**

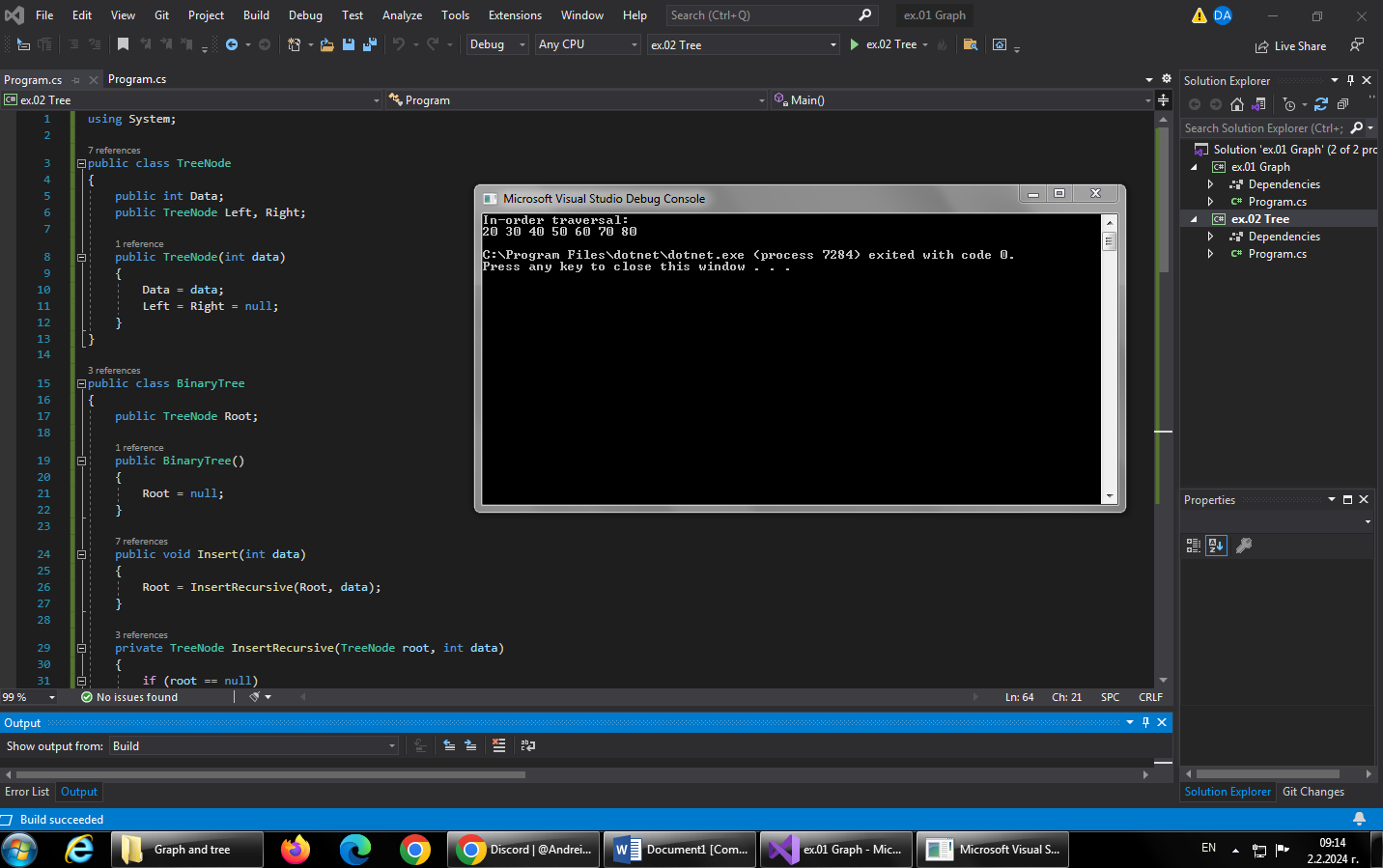
search(X) - търси елемент в дървото

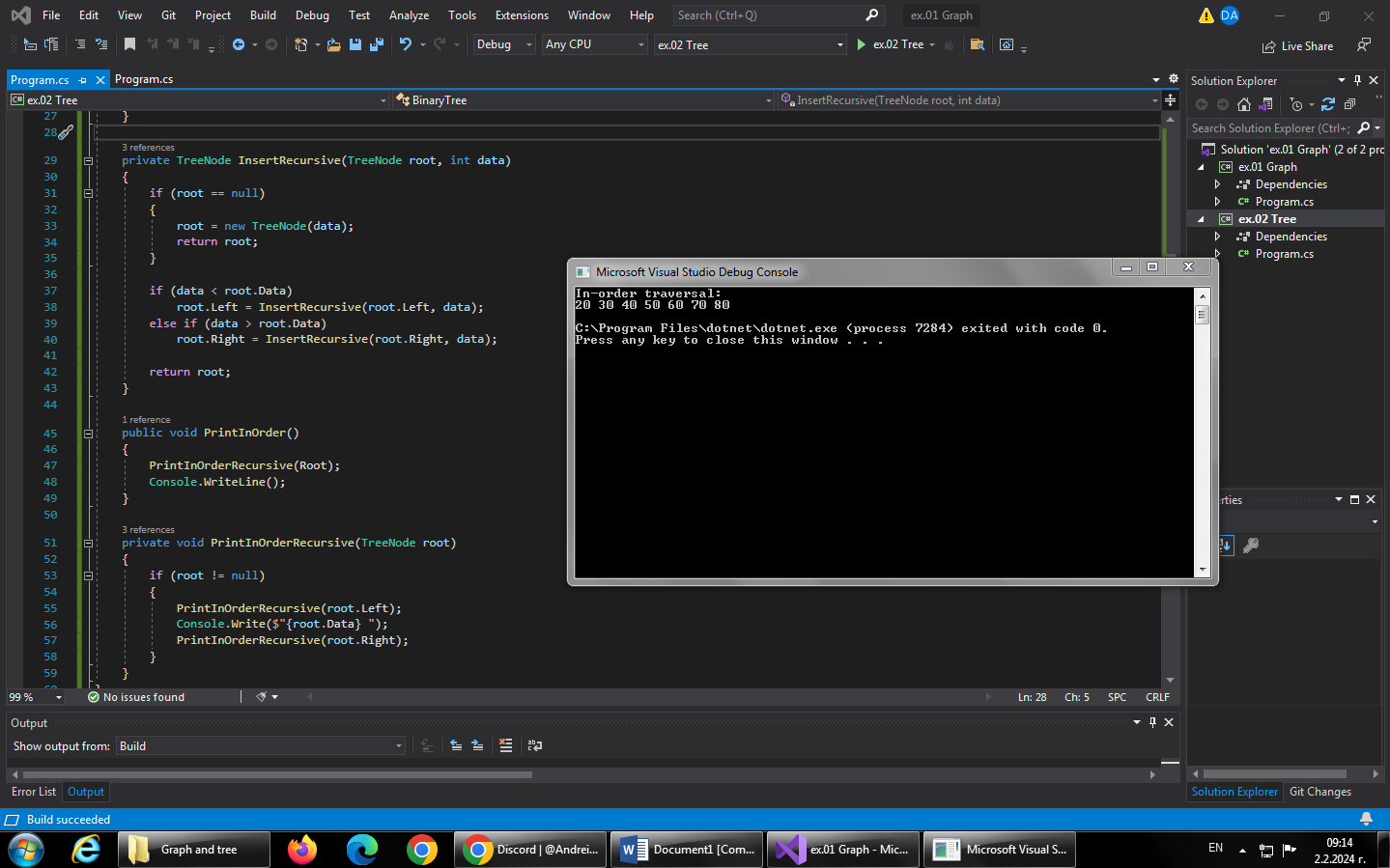
insert(X) - добавя елемент в дървото

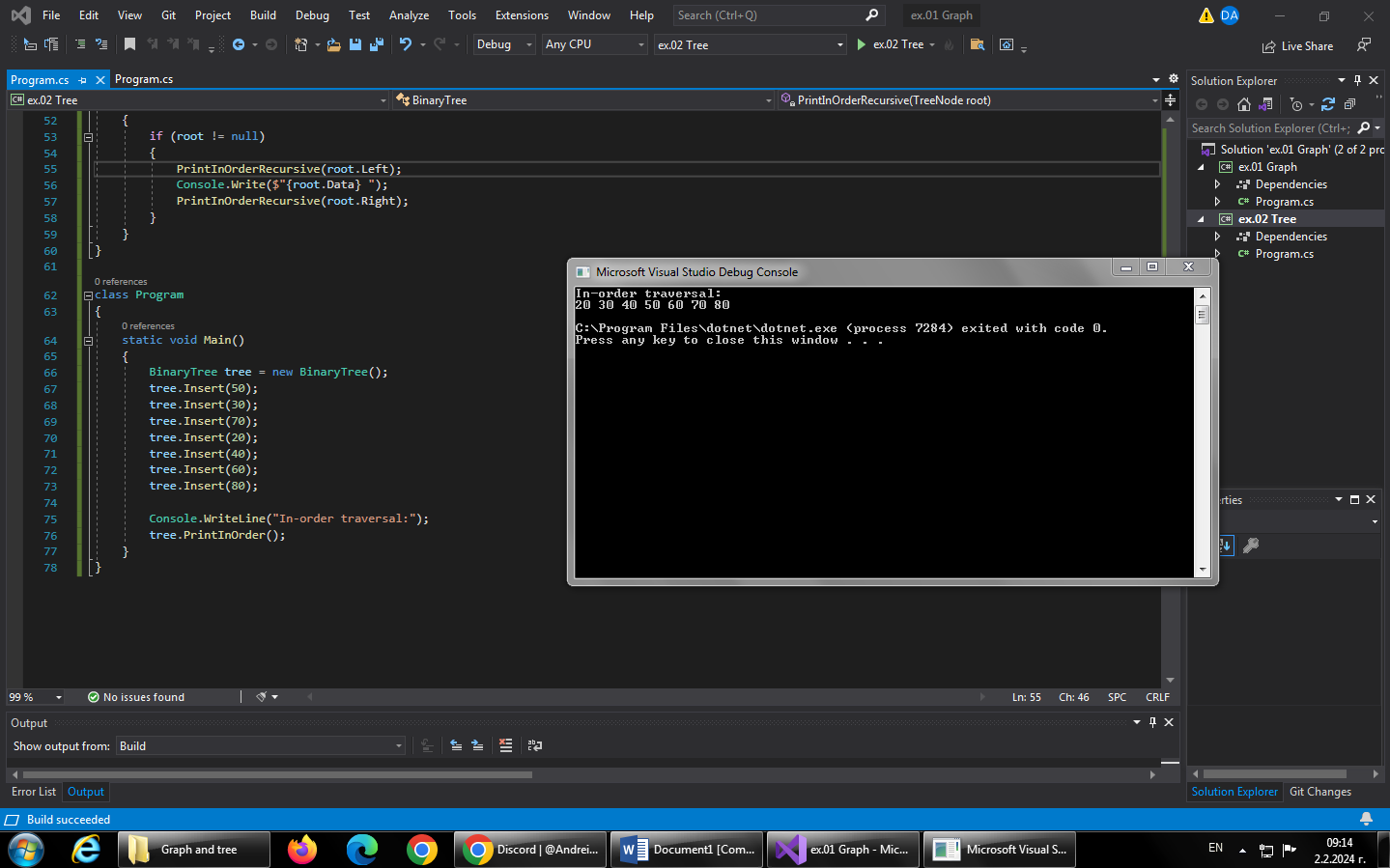
remove(X) - изтрива елемент от дървото

print() - принтира дървото в специфичен ред

**Бинарно дърво в C# - Пример**

****

****

****

Този пример демонстрира създаването на бинарно дърво (BinaryTree), добавяне на върхове и извеждане на върховете в ред на възходящ ред (PrintInOrder).

1. **Увод в теорията на графите**

* Графите се една изключително полезна и доста разпространена структура от данни. Използват се за описването на най-разнообразни взаимовръзки между обекти от практиката, свързани с почти всичко. Както ще видим по-късно, дървета са подмножество на графите, т.е. графите представляват една обобщена структура, позволяваща моделирането на доста голяма съвкупност от реални ситуации.
* Честата употреба на графите в практиката е довела до задълбочени изследвания в "теория на графите", в която са известни огромен брой задачи за графи и за повечето от тях има и добре известно решение.

**Видове представяния:**

* Списък на ребрата – представя се, чрез списък от наредени двойки (vi, vj), където съществува ребро от vi до vj. Ако графът е претеглен, то вместо наредена двойка имаме наредена тройка, като третият ѝ елемент показва какво е теглото на даденото ребро.
* Списък на наследниците – в това представяне за всеки връх v се пази списък с върховете, към които сочат ребрата започващи от v. Тук отново, ако графът е претеглен, към всеки елемент от списъка с наследниците се добавя допълнително поле, показващо цената на реброто до него.
* Матрица на съседство – графът се представя като квадратна матрица g[N][N], в която, ако съществува ребро от vi до vj, то на позиция g[i][j] в матрицата е записано 1. Ако такова ребро не съществува, то в полето g[i][j] е записано 0. Ако графът е претеглен, в позиция g[i][j] се записва теглото на даденото ребро, а матрицата се нарича матрица на теглата. Ако между два върха в такава матрица не съществува път, то тогава се записва специална стойност, означаваща безкрайност.
* Матрица на инцидентност между върхове и ребра – в този случай отново се използва матрица, само че с размери g[M][N], където М е броят на върховете, а N е броят на ребрата. Всеки стълб представя едно ребро, а всеки ред един връх. Тогава в стълба съответстващ на реброто (vi, vj) само и единствено на позиция i и на позиция j ще бъдат записани 1, а на останалите позиции в този стълб ще е записана 0. Ако реброто е примка, т.е. е (vi, vi), то на позиция i записваме 2. Ако графът, който искаме да представим, е ориентиран и искаме да представим ребро от vi до vj, то на позиция i пишем 1, а на позиция j пишем -1.

**Графи – основни операции**

Основните операции в граф са:

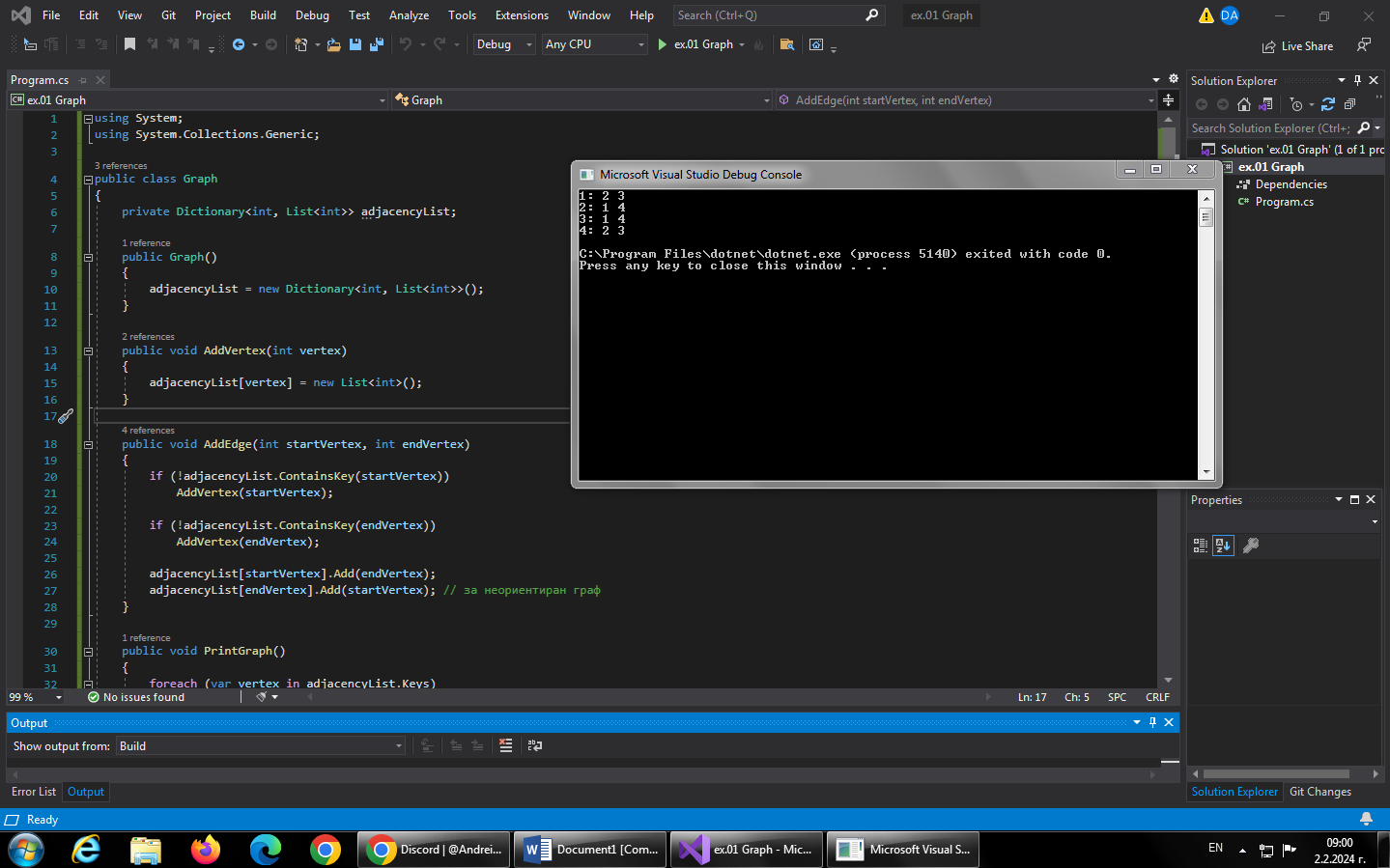
- Създаване на граф

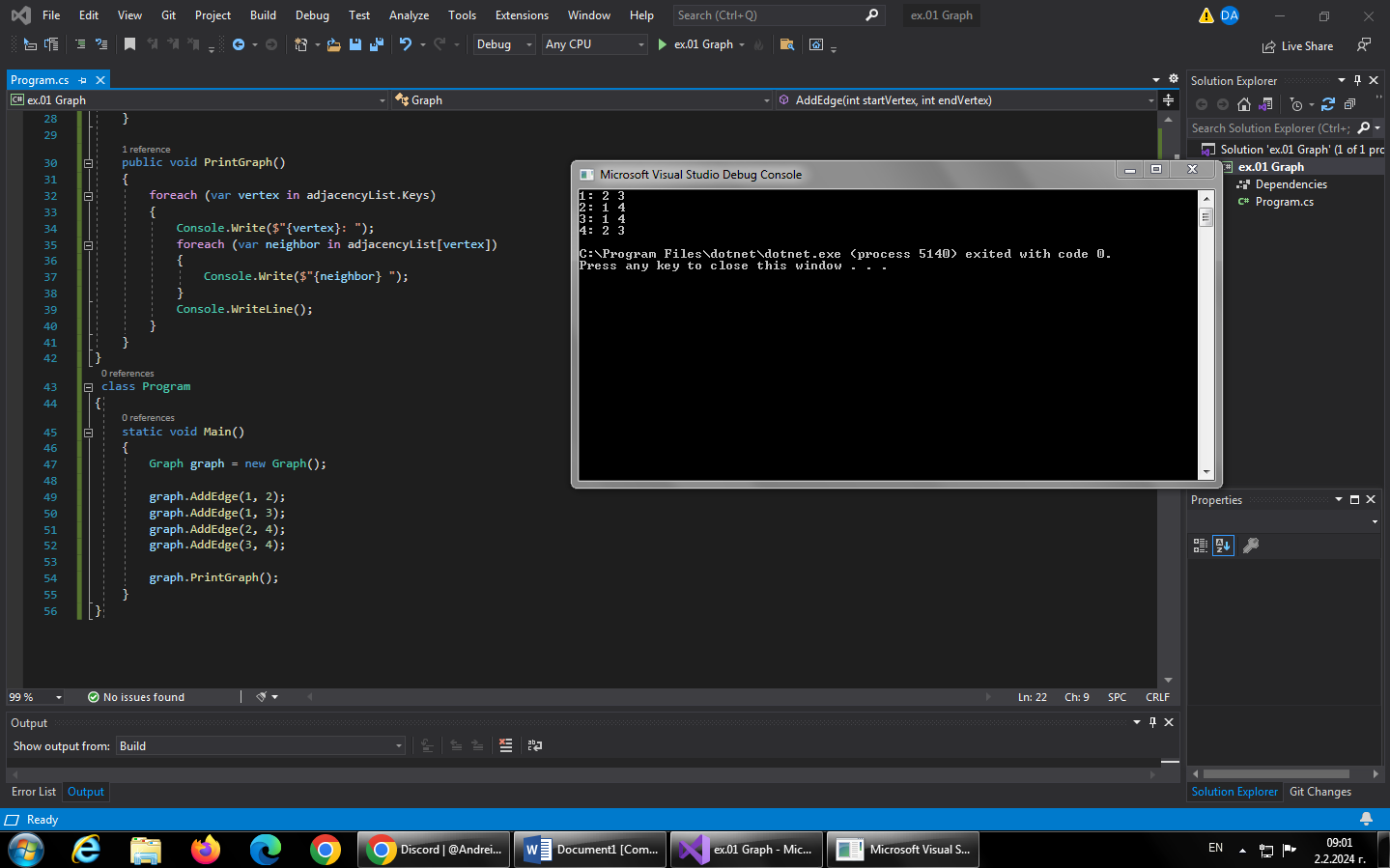
- Добавяне / премахване на връх / ребро

- Проверка дали даден връх / ребро съществува

- Намиране на наследниците на даден връх

**Граф – Пример**

****

****

Този пример създава граф чрез клас `Graph` с методи за добавяне на върхове и ребра. Програмата след това добавя ребра и извежда графа.

1. **Съпоставяне на дърво и граф**

* Дървата и графовете са две различни структури от данни, но имат определени сходства и различия:

**Структура:**

**Дърво:** Има структура от върхове, свързани чрез ребра. Всеки връх (освен корена) има точно един родител. Всеки връх може да има нула или повече наследници.

**Граф:** Също има върхове и ребра, но в графа няма стриктни ограничения за броя на родители или наследници на всяка върхова точка.

**Насоченост:**

**Дърво:** Обикновено е насочено, като има определена посока от корена към листата.

**Граф:** Може да бъде насочен (ориентиран) или неориентиран, като ребрата в ориентиран граф имат посока.

**Цикли:**

**Дърво:** Няма цикли. Всеки връх е достижим точно по един начин.

**Граф:** Може да съдържа цикли, което означава, че може да съществува път, по който можем да обходим върховете и да се върнем към началния.

**Използване:**

**Дърво:** Обикновено се използва за представяне на йерархии и подструктури с четко определени връзки.

**Граф:** Се използва за моделиране на връзките между обекти, където няма стриктна структура на връзките.

* Въпреки тези разлики, дървата са специфичен вид насочен граф, като всеки граф може да бъде ръсен като дърво чрез определени алгоритми за обхождане.

Източници:

<https://www.pg-vpeev.com/wp-content/uploads/2011/12/19-darvo.pdf>

<https://github.com/BG-IT-Edu/School-Programming/blob/main/Courses/Applied-Programmer/Algo-and-Data-Structures-Advanced/05-trees.pdf>

<https://introprogramming.info/intro-csharp-book/read-online/glava17-durveta-i-grafi/>

Изготвил: Валентин Кьосев 12 ,,а‘‘ клас