Мазмұны

[Кіріспе 4](#_Toc184673208)

[1 Пәндік аймақты зерттеу 5](#_Toc184673209)

[1.1 Стек деректер құрылымының негізі 5](#_Toc184673210)

[1.2 Стек тарихы 6](#_Toc184673211)

[1.3 Стектің қолданылу аймағы 7](#_Toc184673212)

[2 Алгоритмдер және математикалық модель 9](#_Toc184673213)

[2.1 Стек құрылымының визуализациясы және оның негізгі ұғымдары 9](#_Toc184673214)

[2.2 Визуализация: Push және Pop алгоритмдері 9](#_Toc184673215)

[3 Алгоритмдер және математикалық модель 13](#_Toc184673216)

[3.1 Қолданылған орта және құралдар 13](#_Toc184673217)

[3.2 Қолданушы интерфейсін жобалау 14](#_Toc184673218)

[3.3 Қолданушы интерфейсін тестілеу 17](#_Toc184673219)

[Қорытынды 22](#_Toc184673220)

[Пайдаланылған әдебиеттер тізімі 23](#_Toc184673221)

[Қосымша А 24](#_Toc184673222)

# Кіріспе

Деректер құрылымдары – қазіргі ақпараттық технологиялардың негізі. Стек деректер құрылымы – ең негізгі әрі кең таралған құрылымдардың бірі. Стек, «соңғы келген – бірінші шығады» (LIFO) принципі бойынша жұмыс істейді, яғни ең соңғы енгізілген элемент бірінші шығарылады. Бұл қарапайым, бірақ тиімді концепция көптеген алгоритмдер мен программаларда қолданылады. Стек жұмысының визуализациясы осы принциптерді түсінуді жеңілдетеді, әсіресе жаңа бастаған бағдарламашылар үшін.

Стек деректер құрылымының тарихы 1950 жылдардың басына оралады, бұл кезең компьютерлік ғылымдардың дамуында үлкен бетбұрыс жасаған уақыт. Алғашқы компьютерлер мен программалау тілдерінің пайда болуы стектің концепциясына негіз болды.

1956 жылы Д. Кнут "The Art of Computer Programming" атты кітабында стек деректер құрылымын алғаш рет сипаттады. Бұл кітапта стек операциялары мен олардың алгоритмдердегі рөлі туралы толық мәлімет берілген. Стек деректер құрылымының кеңінен танылуы 1970 жылдардан басталды, ол кезде программалау тілдерінде рекурсивті функцияларды жүзеге асыру үшін стектер қажет болды.

Стектердің концепциясы компьютерлік архитектураға енгізіліп, процессорлар стектерді функциялар арасындағы контекстті сақтау үшін пайдаланатын болды. Бұл, әсіресе, рекурсивті функциялардың тиімді жұмыс істеуін қамтамасыз етті. Стек концепциясының басқа да қолдану салалары пайда болды, мысалы, веб-браузерлерде пайдаланушының тарихын сақтау үшін стектер пайдаланылады.

Визуализацияның маңызы стектің жұмыс принциптерін, операцияларын және динамикасын көрсету арқылы бағдарламалау мен алгоритмдерді түсінуді жеңілдетуінде жатыр. Визуализаторлар пайдаланушыларға стектің әрекеттерін интерактивті түрде көруге, әртүрлі сценарийлерді сынауға мүмкіндік береді.

Курстық жобаның мақсаты программалық жабдықтама әзірлеу арқылы стектің жұмысын көрсету үшін интерактивті визуализатор жасау.

Курстық жобаның міндеттері:

* теориялық негіздерді зерттей отырып пайдаланушы интерфейсінің ыңғайлылығын қамтамасыз ету;
* алгоритмдерді көрсету және тест жүргізу арқылы кері байланыс жинауды қамтиды;
* интерактивті визуализатор әзірлеу.

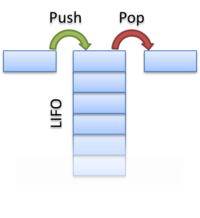
Визуализатордың мақсаты – стектің жұмысын тиімді түрде көрсету, бұл бағдарламалау мен деректер құрылымдарын меңгеруге көмектеседі.

# 1 Пәндік аймақты зерттеу

# 1.1 Стек деректер құрылымының негізі

Стек деректер құрылымы – ақпаратты сақтау мен өңдеудің негізгі тәсілдерінің бірі. Стек LIFO (Last In, First Out) принципімен жұмыс істейді, яғни соңғы енгізілген элемент бірінші болып алынады. Стектің жұмыс істеу принципі қарапайым, бірақ өте тиімді. Элементтер стекке қосылғанда, олар жоғарғы жағында орналасады, ал алынған кезде тек жоғарғы элемент алынады. Стектің мақсаты - деректерді уақытша сақтау және оларға жылдам қол жеткізу.

Стек құрылымының негізі – оның операциялары. Стек операцияларының бастысы push және pop болып табылады. Push операциясы стекке жаңа элементті қосады, ал pop операциясы жоғарғы элементті алып тастайды (Сурет 1.1). Стек элементтері динамикалық немесе статикалық түрде жүзеге асырылуы мүмкін. Динамикалық стек динамикалық жадты, ал статикалық стек алдын ала бөлінген жад аймағын пайдаланады. Динамикалық стектің артықшылығы оның көлемінің өзгеру мүмкіндігі, ал статикалық стек жылдам, бірақ көлемі шектеулі.



Сурет 1.1 – Стек операцияларының қызметі

Стек ережелері өте қарапайым. Тек жоғарғы элементпен жұмыс жасауға болады. Егер стек бос болса, pop операциясы орындалмайды, бұл программаның тұрақтылығын қамтамасыз етеді. Стектің тиімділігі мен қарапайымдылығы оны көптеген алгоритмдер мен программалау тілдерінің негізінде пайдалануға мүмкіндік береді. Стек рекурсивтік функциялардың қайтару адресін сақтау үшін, веб-браузерлерде тарихты басқару үшін, математикалық өрнектерді шешу үшін және графикалық интерфейстерде элементтерді басқару үшін кеңінен қолданылады.

Стек деректер құрылымын қолданудың практикалық мысалдары өте көп. Мысалы, рекурсиялық функциялар шақырылған кезде, әр функция үшін стекке жаңа элемент қосылады. Функция аяқталған кезде, жоғарғы элемент алып тасталады, бұл функцияның қайтару адресін сақтау үшін қажет. Веб-браузерлерде пайдаланушы веб-беттен шыққанда, браузер алдыңғы бетке оралу үшін стек механизмін пайдаланады. Осылайша, стек браузер тарихын басқаруда маңызды рөл атқарады.

Математикалық өрнектерді шешуде де стек пайдалы құрал болып табылады. Инфикс, префикс, және постфикс жазбаларын талдау кезінде стек математикалық операциялардың тәртібін анықтауға көмектеседі. Стек алгоритмдер арқылы деректерді тиімді өңдеу мен анализдеудің негізін құрайды [1].

# 1.2 Стек тарихы

Стек деректер құрылымы 1950-жылдардың соңында пайда болды. Алғашқы стектер функциялардың қайтару адресін сақтау үшін қолданылған. Деректер құрылымдарының дамуы программалау теориясының және математикалық логиканың дамуымен тығыз байланысты болды. Стек деректер құрылымының алғашқы сипаттамалары 1960 жылдары жарияланды, бұл кезеңде стектер әртүрлі алгоритмдер мен программалық жүйелерде кеңінен қолданылды.

1960-1970 жылдары стектердің функциялары мен құрылымы толықтай дамыды. Стек деректер құрылымы математикалық логика мен алгоритмдердің зерттелуімен байланысты болды. Бұл кезеңде стектер көптеген программалау тілдерінде стандартты түрде енгізілді. Pascal, C, және Java сияқты тілдерде стектердің стандартты жүзеге асырылу тәсілдері енгізілді, бұл стектің тиімділігін арттырды.

1980 жылдары программалау тілдерінің дамуы стектердің қолдану аясына жаңаша серпін берді. Стек алгоритмдерін оптимизациялау мен деректерді басқару кезінде пайдалану тиімділігін арттырды. Стектердің пайдаланылуы мен алгоритмдердің жаңартылуы заманауи компьютерлік ғылымдарға елеулі әсер етті.

Стек тарихындағы маңызды аспект - оның алгоритмдерді оңтайландырудағы рөлі. Стектер программалау кезінде функцияларды шақыру мен қайтару механизмдерін тиімді жүзеге асыруға көмектесті. Бұл программаның жылдамдығын арттыру мен ресурстарды үнемдеудің негізгі факторы болып табылады.

Заманауи кезеңде стек деректер құрылымының рөлі әрдайым маңызды болып қалды. Жаңа алгоритмдер мен программалау тілдері пайда болған сайын, стек өзінің орнын сақтап қалды. Стек деректер құрылымының қарапайымдылығы мен тиімділігі программалау және алгоритмдер саласында оны пайдаланудың негізі болып табылады [2].

# 1.3 Стектің қолданылу аймағы

Стек деректер құрылымы әртүрлі салада кеңінен қолданылады. Бағдарламалауда стек функциялар арасындағы байланыстарды басқару үшін маңызды рөл атқарады. Рекурсивтік функцияларды жүзеге асыру кезінде стек деректерінің сақталуы өте маңызды, өйткені әрбір функция шақыруы үшін қайтару адресін сақтау қажет. Стек рекурсиямен жұмыс істегенде функцияның қайтару нүктесін сақтау үшін қажет.

Веб-браузерлерде стектер пайдаланушының тарихын сақтау үшін қолданылады. Мысалы, пайдаланушы веб-беттен кеткенде, браузер алдыңғы бетке оралу үшін стек механизмін пайдаланады. Бұл механизм браузердің интерфейсінің бір бөлігін құрайды және пайдаланушыға оңай навигация жасауға мүмкіндік береді.

Сонымен қатар, математикалық өрнектерді шешу, мысалы, инфикс, префикс, және постфикс жазбаларын анализдеу стектің негізгі қолдану аймақтарының бірі болып табылады. Бұл жағдайларда стек математикалық операциялардың тәртібін анықтауға көмектеседі, күрделі өрнектерді жеңілдетуге мүмкіндік береді.

Компьютерлік графикада стек сцена элементтерінің бөлінуі мен дәйектемелері кезінде өте пайдалы. Стек, сондай-ақ, операциялық жүйелерде процессордың контекстін сақтау үшін, сонымен қатар қатты диск операцияларында файлдардың жағдайын бақылау үшін де қолданылады.

Стек деректер құрылымы жасанды интеллект пен машиналық оқытуда да пайдаланылады. Мысалы, нейрондық желілерді құру кезінде стек алгоритмдерін пайдалануға болады. Стек алгоритмдері, сонымен қатар, мәліметтерді өңдеу мен анализдеу кезінде тиімді шешімдер ұсынады.

Сонымен қатар, ойын бағдарламалауында стек деректер құрылымының қолданылуы кеңінен таралған. Ойынның әрбір деңгейі немесе күйі стекке сақталады, бұл ойыншының прогресін бақылауға мүмкіндік береді. Стек көмегімен ойынның жағдайын қалпына келтіру, яғни ойыншының алдыңғы күйіне оралуы оңай жүзеге асырылады [3].

Қорытындылай келе стек деректер құрылымы ақпаратты ұйымдастыру мен өңдеудің негізгі тәсілдерінің бірі болып табылады. LIFO принципімен жұмыс істейтін стек программалауда, алгоритмдерде және әртүрлі қосымшаларда кеңінен қолданылады. Оның қарапайымдылығы мен тиімділігі, әсіресе, функциялар арасындағы байланыстарды басқару мен рекурсивтік шақыруларды жүзеге асыруда маңызды рөл атқарады.

Стектің тарихы 1950-жылдардан басталып, уақыт өте келе программалау тілдері мен алгоритмдердің дамуына үлкен ықпал етті. Стек деректер құрылымының эволюциясы жаңа алгоритмдер мен программалау парадигмаларының пайда болуына себепші болды, бұл өз кезегінде стектің танымалдылығын арттырды.

Стек деректер құрылымы түрлі салаларда, соның ішінде веб-браузерлерде, компьютерлік графикада, ойын бағдарламалауда және математикалық өрнектерді шешуде маңызды рөл атқарады. Оның пайдалану аясы кең, ал практикада тиімділігі мен қарапайымдылығы оны программисттер мен зерттеушілер арасында жоғары бағалатады.

Осылайша, стек деректер құрылымы компьютерлік ғылымдар мен информатикадағы маңызды элемент болып табылады. Ол программалау барысында тиімді шешімдер ұсынуға, ресурстарды басқаруға және деректермен жұмыс істеуді жеңілдетуге мүмкіндік береді. Стек алгоритмдерін түсіну мен пайдалану программисттерге тиімді код жазуға және алгоритмдердің тиімділігін арттыруға көмектеседі.

# 2 Алгоритмдер және математикалық модель

# 2.1 Стек құрылымының визуализациясы және оның негізгі ұғымдары

Бұл бөлімде cтек құрылымы деректерді сақтау үшін LIFO (Last In, First Out – соңғы кірген бірінші шығады) принципіне негізделген. Бұл құрылымда деректердің қосылуы мен алынуы тек бір жағынан, яғни стек жоғарғы жағынан ғана жүзеге асырылады. Әрбір жаңа элемент стектің жоғарғы жағына қосылады, ал элементтер тек жоғарғы жағынан алынады. Стек операцияларын визуализациялау арқылы оның жұмысын түсіну әлдеқайда жеңілдейді. Визуализация – стектің динамикалық өзгерістерін көрнекі түрде көрсету, яғни деректердің стектен қосылуы мен шығарылуын анимациялар немесе графиктер арқылы көрсету.

Стекке элемент қосу (Push) немесе алу (Pop) кезінде стектің ішіндегі элементтер орын ауыстырады. Бұл өзгерістердің әрбір қадамын көру, әсіресе бағдарламалау мен деректер құрылымдарын оқыту кезінде, өте маңызды. Стек визуализациясында әрбір жаңа элемент стектің жоғарғы жағына қосылып, ал pop операциясы орындалғанда, жоғарғы элемент алынып тасталады. Бұл әрекеттер көру арқылы жүзеге асырылғанда, студенттер мен бағдарламашылар стектің жұмысын нақты көріп, түсіне алады.

Стек құрылымының визуализациясы көбінесе динамикалық түрде көрсетіледі, мысалы, графикалық интерфейстер немесе анимациялар арқылы. Бұл құралдар арқылы әрбір push және pop операциясының қалай жүзеге асатынын көруге болады. Стекке жаңа элемент қосылғанда, ол жоғары қарай қозғалып, ал pop операциясы орындалған кезде, жоғарғы элемент өшіріледі. Бұл әрекеттер элементтердің қалай сақталып, қалай алынып тасталғанын көрсетеді.

Визуализация кезінде стектің ішіндегі элементтердің орын ауыстыруы нақты уақыт режимінде көрінеді. Әрбір жаңа элемент қосылғанда немесе жойылғанда, бұл өзгерістер айқын көрініс табады, және оларды графикалық түрде ұсыну арқылы алгоритмдер мен деректер құрылымдарын түсінуге болады. Сонымен қатар, визуализация алгоритмдерді орындау барысында қандай өзгерістер болып жатқанын нақты көруге мүмкіндік береді. Бұл әсіресе рекурсиялы алгоритмдер мен терең іздеу алгоритмдерінде маңызды, өйткені олар да стек құрылымын пайдаланады [4].

# 2.2 Визуализация: Push және Pop алгоритмдері

Push, Pop операциялары стектің негізгі әрекеттері болып табылады, және оларды визуализациялау стектің қалай жұмыс істейтінін түсінуге көмектеседі (Cурет 2.1).

Push операциясы – жаңа элементті стектің жоғарғы жағына қосу. Бұл операция орындалған кезде стектің жоғарғы жағына жаңа элемент қосылады, ал визуализация кезінде жаңа элементтің қосылуы көрсетіледі. Әдетте, жаңа элемент жоғарғы жағына көтеріледі, ал стектің биіктігі артады. Мысалы, егер бастапқыда стек бос болса, бірінші элемент қосылғанда ол стектің бірден бірінші элементі болады. Келесі элемент қосылған кезде, ол жоғарғы жағына орналасып, стек биіктігін арттырады.

Push операциясын визуализациялағанда, элементтің қалай қосылып жатқанын көрсету үшін анимация немесе графиктер қолданылады. Әрбір жаңа элементтің қосылуы стектің жоғарғы жағына қарай жылжиды, сондықтан визуализацияда жаңа элементтің пайда болуы мен оның орналасуы көрсетіледі. Бұл студенттерге стектің жұмысын көрнекі түрде түсінуге мүмкіндік береді.

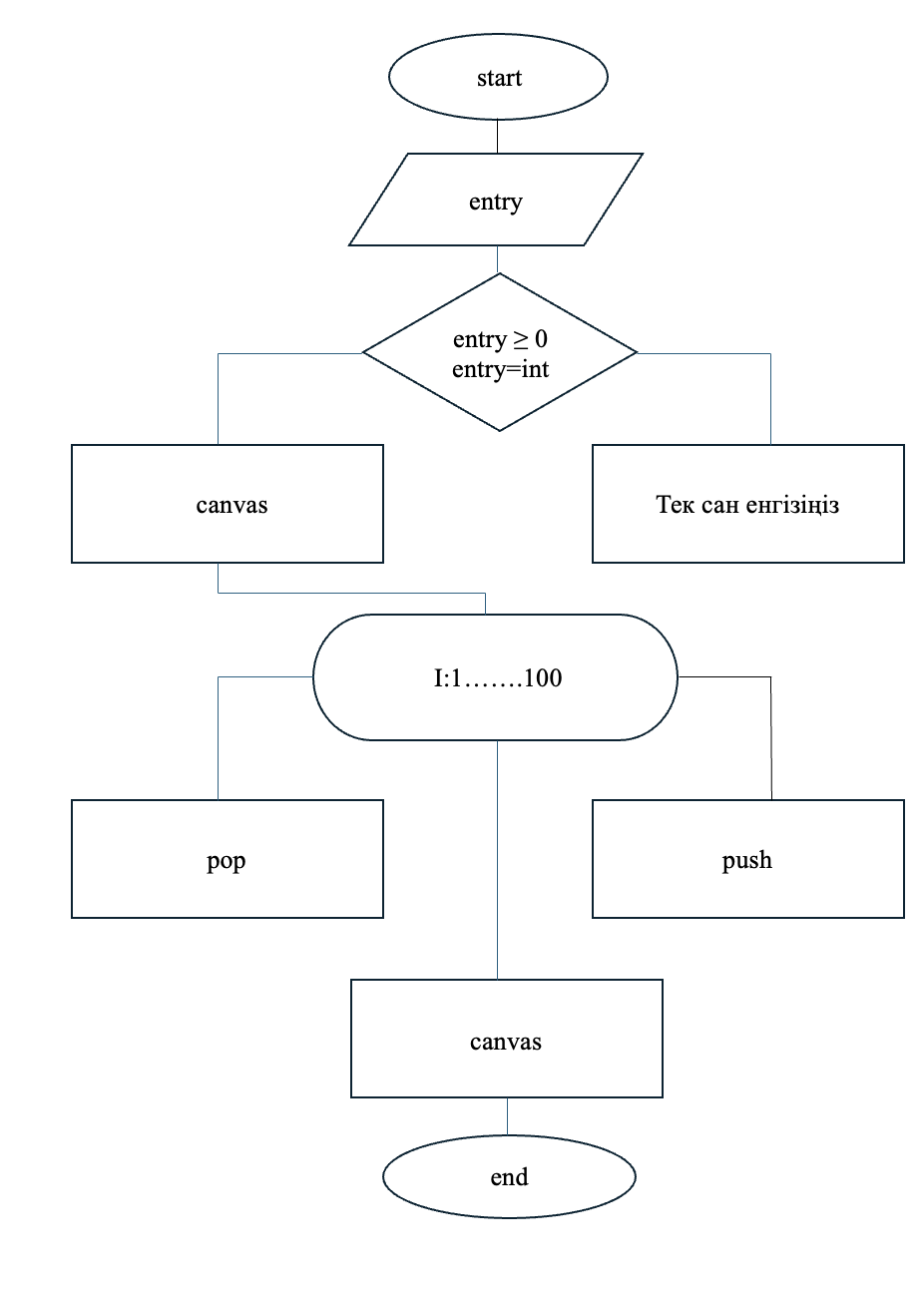
Pop операциясы – стектің жоғарғы элементін алып тастау. Стек жоғарғы жағындағы элементті алып тастаған кезде, стектің биіктігі бірлікке төмендейді. Визуализация барысында pop операциясы орындалған кезде жоғарғы элементтің өшірілуі көрсетіледі, және бұл процесс анимация арқылы көрсетіледі. Әрбір pop операциясы орындалған сайын, стектің жоғарғы жағы төмендеп, соңғы элементтің өшірілуі көрсетіледі.

Pop операциясы орындалғанда, жаңа жоғарғы элемент төменде орналасқан элементтерден біреу болуы мүмкін, және оның орны стектің жоғарғы жағына көтеріліп, жаңа жоғарғы элемент ретінде көрсетіледі. Бұл визуализацияның арқасында, стектің динамикалық жұмысын түсіну оңай болады.

Peek немесе Pop операциясы – стектің жоғарғы элементін көру. Бұл операция стектің ішіндегі деректерді тексеру үшін қолданылады, бірақ элементті алып тастамайды. Peek операциясы визуализация кезінде стектің жоғарғы элементін көрсетеді, бірақ ол ешқандай өзгеріссіз қалады. Бұл операция, әсіресе деректерді тексеру кезінде пайдалы, бірақ стектің биіктігі өзгермейді.

Peek операциясының визуализациясы элементтің тек көрсетілуін ғана қамтиды, ал қалған элементтер орнында қалады. Бұл элементтің өзгермейтінін көру арқылы, пайдаланушы стектің ішкі жағдайын біле алады.

Push, Pop және Peek операцияларының барлығы деректер құрылымдарын тиімді пайдалану үшін маңызды. Визуализация арқылы бұл операциялардың қалай орындалатынын нақты көруге болады, ал бұл бағдарламалау мен алгоритмдер туралы түсінікті айтарлықтай тереңдетеді.



Сурет 2.1 – Алгоритмдер және математикалық модель

Бағдарлама іске қосылған кезде қолданушыдан матрицаның өлшемі сұралады, яғни жолдар мен бағандардың саны. Қолданушы енгізген мәліметтер бойынша бағдарлама кездейсоқ сандармен толтырылған матрица құрады. Әрбір элемент кездейсоқ санмен генерацияланып, олар матрицаға орналасады. Бұл элементтер экранда тікбұрыштар түрінде бейнеленіп, бірінің үстіне бірі орналастырылады.

Осыдан кейін бағдарлама қолданушының әрекетін күтуге көшеді. Қолданушы «Push» немесе «Pop» батырмаларын басуы мүмкін. Егер «Push» батырмасы басылса, матрицаның соңына жаңа элемент қосылады. Бұл элемент кездейсоқ сандармен толтырылады. Егер «Pop» батырмасы басылса, матрицаның соңғы элементі жойылады.

Әрбір әрекеттен кейін матрица жаңартылады: жаңа элемент қосылғанда немесе жойылғанда, экрандағы көрініс жаңарып отырады. Осылайша, қолданушы матрицаны динамикалық түрде басқаруға мүмкіндік алады.

Бағдарлама бірнеше рет «Push» және «Pop» әрекеттерін орындау мүмкіндігін береді. Бағдарламаны тоқтату үшін қолданушы арнайы батырманы басуы қажет, сонымен бірге барлық ресурстар босатылып, бағдарлама жабылады.

Бұл алгоритм қолданушының матрицамен жұмыс істей отырып, оның элементтерін қосуға және жоюға мүмкіндік береді, ал барлық өзгерістер экранда көрініс табады.

# 3 Алгоритмдер және математикалық модель

# 3.1 Қолданылған орта және құралдар

Жобаны әзірлеу барысында программалау ортасы ретінде VS Code қолданылады, ал программалау құралдары ретінде Python және Tkinter пайдаланылады. Нұсқаларды басқару үшін git, ал ортаны баптау үшін venv қолданылды.

VS Code (Visual Studio Code) – бұл Microsoft компаниясы әзірлеген, мәтін мен код редакторы болып табылатын жеңіл және функционалды құрал. Ол әртүрлі бағдарламалау тілдерінде жұмыс істеуге мүмкіндік беретін, кеңейтуге және баптауға болатын кросс-платформалы (Windows, macOS, Linux) орта [5].

Git – файлдардың өзгерістерін басқаруға арналған нұсқаларды басқару жүйесі. Ол бағдарламалық жасақтама жобаларында өзгерістер тарихын қадағалап, командалық жұмысты ұйымдастыруға мүмкіндік береді. Git көмегімен бірнеше адам бір уақытта бір файлда жұмыс істей алады және өзгерістердің қайсысы қашан және кім тарапынан жасалғанын көруге болады [6].

Git үлестірілген жүйе болып табылады, яғни әрбір пайдаланушы репозиторийдің толық көшірмесін өз құрылғысында сақтайды. Бұл жүйенің басты ерекшелігі – желіге тәуелсіз жұмыс істей алуы және деректердің қауіпсіздігі. Жұмыс үш негізгі аймақта жүзеге асады: жұмыс аймағы, индексация аймағы және жергілікті репозиторий. Жұмыс аймағында файлдар өзгертіледі, индексация аймағына commit жасалатын өзгерістер таңдалады, ал репозиторийде барлық нұсқалардың тарихы сақталды.

Git файлдарды қосу, өзгерту немесе жою секілді барлық операцияларды тіркеп отырады. Сонымен қатар, алдыңғы нұсқаларға оралуға, параллель тармақтарды (branch) құруға және біріктіруге мүмкіндік береді. Бұл оның икемділігі мен командалық жұмысқа тиімділігін қамтамасыз етеді.

Tkinter – Python тілінде графикалық интерфейстер (GUI) құру үшін қолданылатын стандартты кітапхана. Ол Python-ның стандартты модульдеріне кіреді және кросс-платформалы, яғни Windows, macOS, және Linux жүйелерінде жұмыс істей алады. Tkinter графикалық интерфейстерді құруға арналған «Tk» құралын пайдаланады [7].

Tkinterдің жұмыс істеу принципі оқиғаларға негізделген бағдарламалау әдісіне сүйенеді. Қолданушы интерфейспен әрекеттескен сайын (батырманы басу, мәтін енгізу, мәзірді таңдау), белгілі бір оқиғалар туындайды. Бағдарлама бұл оқиғаларды арнайы жазылған функциялар арқылы өңдейді.

Tkinter арқылы интерфейстің әртүрлі элементтерін жасауға болады: батырмалар, мәтіндік өрістер, мәзірлер, чекбокстар, радио түймелер, холсттар және басқа виджеттер. Бұл элементтерді басқару үшін арнайы әдістер мен қасиеттер қолданылады.

Бағдарламада Tkinter қолдану үшін алдымен терезені құрып, оған қажетті виджеттерді орналастыру қажет. Кейін негізгі цикл (mainloop) іске қосылады, бұл бағдарлама жұмысы аяқталғанша оқиғаларды күтіп, өңдеп отырады [8].

Venv – Python тілінде виртуалды орта құруға арналған құрал. Виртуалды орта – белгілі бір Python жобасы үшін тәуелсіз жұмыс істейтін, оқшауланған орта. Ол жобаның қажетті кітапханаларын басқа жобалардан тәуелсіз басқаруға мүмкіндік береді.  
 Venv жаңа виртуалды орта құрған кезде, Python-ның стандартты кітапханаларының көшірмесін жасайды және жоба үшін бөлек тәуелділіктер орнатуға мүмкіндік береді. Бұл ортаның өз каталогы болады, онда Python интерпретаторы мен кітапханалар сақталады.

Виртуалды ортаны қолдану үшін оны белсендіру қажет. Белсендіру кезінде командалық жол автоматты түрде осы виртуалды ортаға сілтейді. Бұл тәуелділіктерді орнату немесе қолдану кезінде Python-ның негізгі орнатылуына әсер етпейді [9].

Venv-ті қолданудың басты артықшылығы – әртүрлі жобаларда әртүрлі Python нұсқаларын немесе кітапханалардың әртүрлі нұсқаларын қолдану мүмкіндігі. Бұл кітапханалардың үйлесімсіздігі мәселелерін болдырмайды және жобаның көшіруін немесе орналастыруын жеңілдетеді.

Виртуалды ортаны құру үшін python «-m venv <орта\_аты>» командасы қолданылады, ал оны белсендіру немесе өшіру үшін арнайы командалар орындалады (жүйеге байланысты) [10].

# 3.2 Қолданушы интерфейсін жобалау

Бұл бөлімде Python бағдарламалау тіліндегі Tkinter кітапханасының көмегімен стектермен жұмыс істеуге арналған қарапайым интерфейс құру үдерісі сипатталады. Бұл интерфейсті құрудың негізгі мақсаты – деректерді стекте сақтау, оларды қосу және жою операцияларын орындауға мүмкіндік беру, сонымен қатар қолданушыларға осы әрекеттерді визуализациялау арқылы түсінікті түрде көрсету.

Қолданушы интерфейсі екі негізгі терезеден тұрады. Негізгі терезе арқылы қолданушы сан енгізіп, оны стекте жаңа жол ретінде сақтай алады. Бұл терезеде мәтіндік жапсырма, енгізу өрісі және екі батырма бар. Бірінші батырма енгізілген санды негізге ала отырып стекте жаңа жол қосу үшін қолданылады. Екінші батырма екінші терезені ашып, стектің мазмұнын визуализациялауға мүмкіндік береді.

Екінші терезе стекті басқару және оның мазмұнын графикалық түрде көрсету үшін арналған. Бұл терезеде стектегі мәліметтерді графикалық түрде көрсету үшін Canvas элементі қолданылады. Сонымен қатар, Push және Pop батырмалары арқылы жаңа элементтер қосуға немесе бар элементтерді алып тастауға болады. Терезеде стектің ағымдағы күйі мәтін түрінде де көрсетіледі(Листинг 3.1).

Стекпен жұмыс істеу барысында екі негізгі операция орындалады: Push және Pop. Push операциясы арқылы қолданушы жаңа элементті немесе жолды стекте қосады. Егер стек бос болса, жаңа жол бірінші ретінде қосылады. Pop операциясы стектен соңғы элементті немесе жолды жою үшін қолданылады. Егер жол бос қалса, ол толықтай стектен алынады. Бұл операциялар стектің «соңғы кірген – бірінші шығады» қағидасына сәйкес жұмыс істейді.

Интерфейс қолданушы тәжірибесін жақсарту мақсатында бірнеше маңызды мүмкіндіктерді қамтиды. Мысалы, қолданушы тек сандарды енгізе алады, ал басқа мәліметтерді енгізу кезінде қате туралы хабарлама шығады. Әрбір әрекеттен кейін арнайы хабарламалар арқылы оның нәтижесі көрсетіледі. Сондай-ақ, барлық өзгерістер нақты уақыт режимінде визуализацияланып, қолданушыға деректер құрылымын жақсырақ түсінуге көмектеседі.

Жалпы алғанда, интерфейс қарапайым әрі интуитивті. Ол қолданушыларға стектермен жұмыс істеудің негізгі операцияларын меңгеруге, сондай-ақ деректерді графикалық түрде көруге мүмкіндік береді. Мұндай интерфейстер деректер құрылымдарын оқытуда немесе шағын бағдарламаларды құруда қолдануға ыңғайлы.

Листинг 3.1 – Қарапайым интерфейс құру үдерісі

import tkinter as tk

from tkinter import messagebox

import random

stack = []

root = tk.Tk()

root.title("Енгізу терезесі")

root.geometry("700x700")

root.config(bg="lightblue")

label1 = tk.Label(root, text="Сан енгізіңіз", font=("Times New Roman", 16, "bold italic"))

label1.pack(pady=50)

def open\_second\_window():

second\_window = tk.Toplevel(root)

second\_window.title("Жұмыс істеу терезесі")

second\_window.geometry("700x700")

label = tk.Label(second\_window, text="Жұмыс істеу терезесі")

label.pack(pady=20)

canvas = tk.Canvas(second\_window, width=500, height=400, bg="white")

canvas.pack(pady=10)

def update\_canvas():

canvas.delete("all")

rect\_width = 80

rect\_height = 30

y\_offset = 400

for i, row in enumerate(reversed(stack)):

for j, value in enumerate(row):

x1 = 200

x2 = x1 + rect\_width

y1 = y\_offset - rect\_height

y2 = y\_offset

canvas.create\_rectangle(x1, y1, x2, y2, fill="skyblue", outline="black")

canvas.create\_text((x1 + x2) // 2, (y1 + y2) // 2, text=str(value), font=("Arial", 12))

y\_offset = y1

matrix\_text = "\n".join(str(row) for row in stack)

matrix\_label = tk.Label(second\_window, text="Матрица:\n" + matrix\_text)

matrix\_label.pack(pady=10)

def push():

if stack:

new\_value = random.randint(1, 100)

stack[-1].append(new\_value)

messagebox.showinfo("Успех", f"Қосылған элемент: {new\_value}")

else:

new\_value = random.randint(1, 100)

stack.append([new\_value])

messagebox.showinfo("Успех", f"Создана новая строка с элементом: {new\_value}")

update\_canvas()

def pop():

if stack:

if stack[-1]:

removed\_value = stack[-1].pop()

messagebox.showinfo("Успех", f"Өшірілген элемент: {removed\_value}")

if not stack[-1]:

stack.pop()

else:

messagebox.showerror("Ошибка", "Матрица пуста! Нечего удалять.")

update\_canvas()

button\_push = tk.Button(second\_window, text="Push", command=push, font=("Times New Roman", 14, "bold italic"))

button\_push.pack(pady=10)

button\_pop = tk.Button(second\_window, text="Pop", command=pop, font=("Times New Roman", 14, "bold italic"))

button\_pop.pack(pady=10)

update\_canvas()

entry = tk.Entry(root, font=("Arial", 14))

entry.pack(pady=10)

def add\_row\_to\_stack():

entered\_text = entry.get()

if entered\_text.isdigit():

cols = int(entered\_text)

row = [random.randint(1, 100) for \_ in range(cols)]

stack.append(row)

messagebox.showinfo("Успех", f"Қосылған матрица: {row}")

else:

messagebox.showerror("Қате", "Тек сан енгізіңіз!")

button\_submit = tk.Button(root, text="Қосылатын элементтер", command=add\_row\_to\_stack, font=("Times New Roman", 14, "bold italic"))

button\_submit.pack(pady=10)

button\_visualize = tk.Button(root, text="Визуализация", command=open\_second\_window, font=("Times New Roman", 14, "bold italic"))

button\_visualize.pack(pady=10)

root.mainloop()

# 3.3 Қолданушы интерфейсін тестілеу

Ең алдымен, тестілеу енгізу өрісінен басталады. Бағдарлама тек сандарды енгізуге мүмкіндік береді, сондықтан осы талаптың орындалуын тексеру қажет. Егер қолданушы сандардан басқа мәтін немесе арнайы символдар енгізсе, бағдарлама қате туралы хабарлама көрсетуі тиіс (Сурет 3.2). Бұл жағдай дұрыс өңделгенін тексеру үшін енгізу өрісіне әртүрлі мәліметтер енгізу керек, мысалы, әріптер, бос орындар немесе аралас символдар.

Келесі қадамда «Қосу» батырмасын тексеру жүзеге асырылады. Бұл батырма енгізу өрісіндегі санға сәйкес стекте жаңа жол құруы тиіс. Егер сан енгізілсе, жаңа жол кездейсоқ сандардан тұрып, стекте сақталуы керек. Сонымен қатар, қолданушыға жаңа жолдың қосылғаны туралы хабарлама көрсетіледі. Осы кезеңде бірнеше сандарды енгізіп, олардың стекте дұрыс сақталып жатқанын бақылау маңызды.

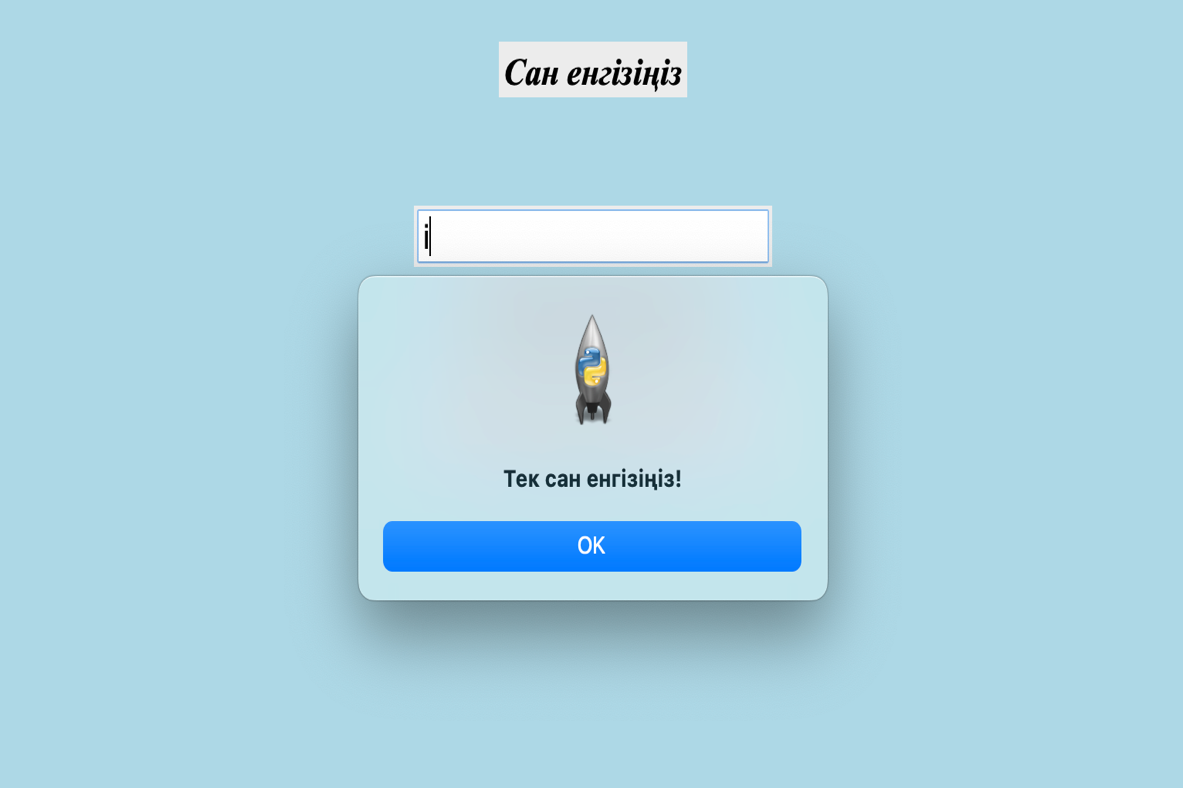
«Визуализациялау» батырмасын тексеру үшін оны басқанда екінші терезе ашылатынына және оның мазмұны дұрыс көрсетілетініне көз жеткізу керек (Сурет 3.3). Екінші терезе ашылғанда, Canvas элементі арқылы стектегі мәліметтер графикалық түрде көрсетілуі тиіс. Бұл жерде әрбір элементтің дұрыс позицияда орналасқанын және графикалық элементтердің бір-бірімен сәйкес келетінін тексеру қажет (Сурет 3.4).

Push және Pop батырмаларын тексеру стектің негізгі операцияларын орындау мүмкіндігін бағалайды. Push батырмасы соңғы жолға жаңа элемент қосуы тиіс (Сурет 3.5). Бұл операция бірнеше рет қайталанып, стектің соңғы жолындағы элементтердің ұлғаюы бақыланады. Pop батырмасы соңғы элементті жоюға мүмкіндік береді (Сурет 3.6). Егер соңғы жол бос қалса, бұл жол автоматты түрде жойылуы керек. Сонымен қатар, егер стек бос болса және Pop батырмасын басқан жағдайда, бағдарлама қолданушыға қате туралы хабарлама көрсетуі тиіс.

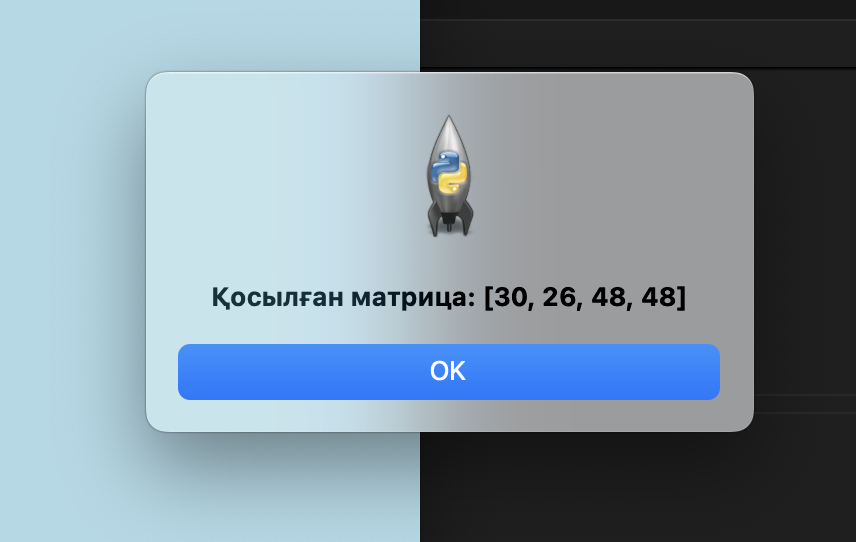
Бағдарламаның шектік жағдайларда жұмыс істеу қабілеті де тексеріледі. Мысалы, үлкен көлемдегі мәліметтермен жұмыс істеу кезінде интерфейстің дұрыс жұмыс істейтініне көз жеткізу керек. Сонымен қатар, визуализацияның сапасы үлкен мәліметтер үшін де сақталуы тиіс. Бұл кезеңде әртүрлі көлемдегі мәліметтермен жұмыс істеп, Canvas элементінің толық деректерді көрсете алатынын тексеру маңызды (Сурет 3.7).

Тестілеудің соңғы қадамында интерфейстің қолданушыға ыңғайлылығы бағаланады. Бағдарламаның түсінікті екеніне, элементтердің орналасуының дұрыс ұйымдастырылғанына көз жеткізіледі. Қолданушылардың қателіктерін дұрыс өңдеу, кері байланыстың жылдам берілуі және деректерді визуализациялау сапасы ерекше назарға алынады.

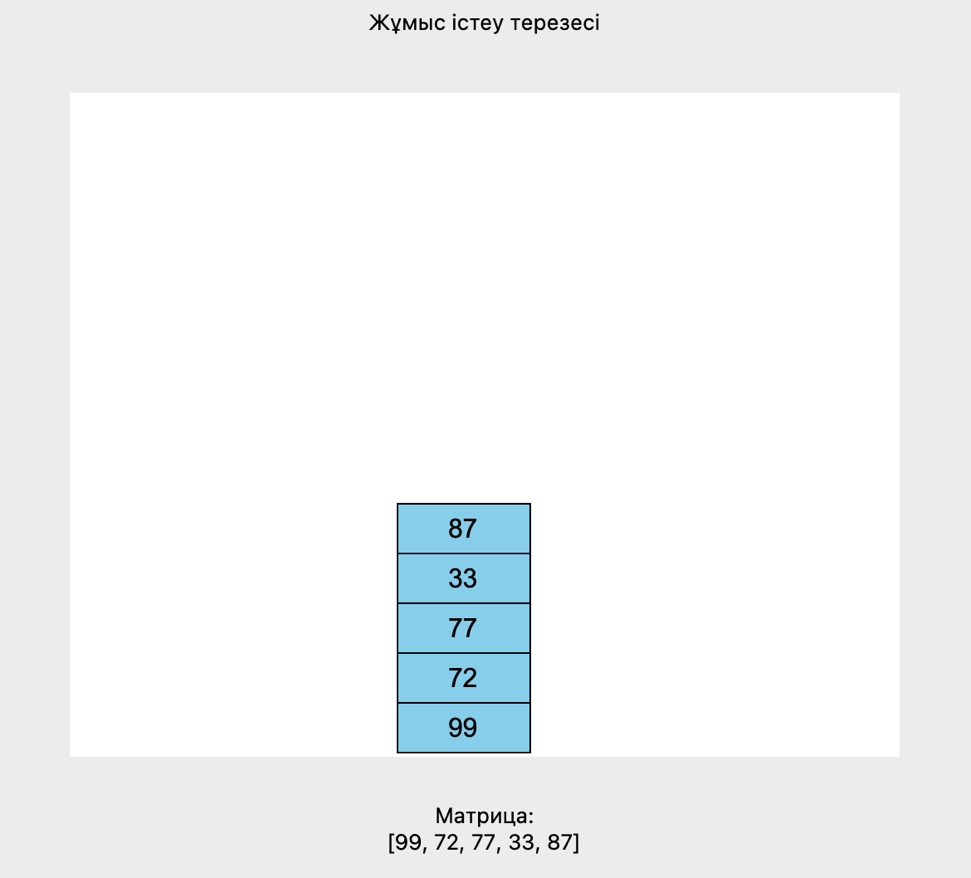
Қорытындылай келе, тестілеу нәтижесінде бағдарламаның барлық негізгі және қосымша функцияларының дұрыс жұмыс істейтіні тексерілді. Қолданушы интерфейсі интуитивті, сенімді және қолдануға ыңғайлы екеніне көз жеткізіледі. Тестілеу бағдарламаның функционалдығын жетілдіруге және оның тұрақтылығын қамтамасыз етуге мүмкіндік береді.



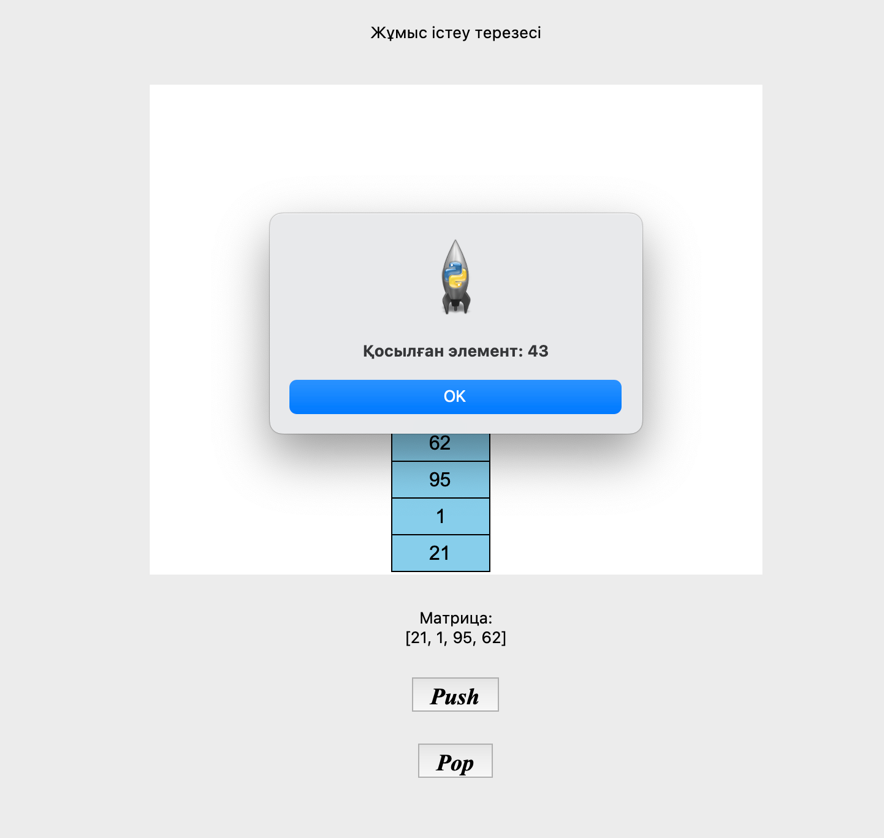
Сурет 3.2 – Қате туралы хабарлама



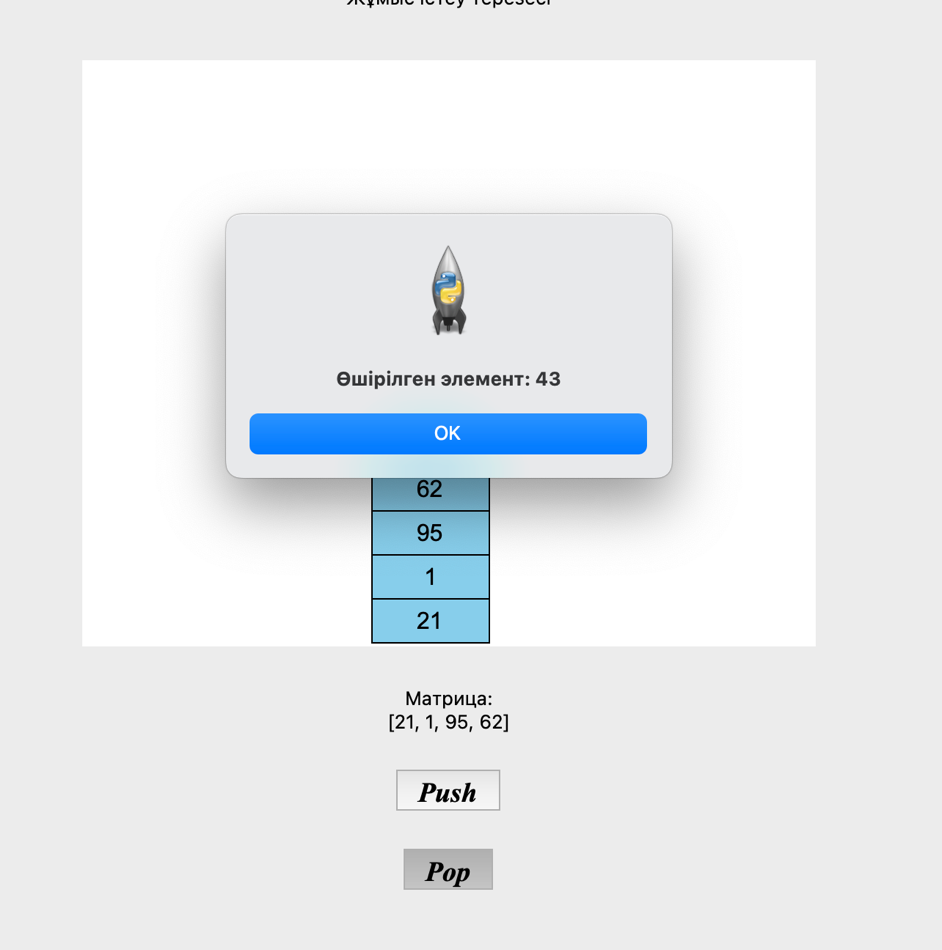
Сурет 3.3 – Cтектегі мәліметтер



Сурет 3.4 – Графикалық элементтер



Сурет 3.5 – Push батырмасы



Сурет 3.6 – Pop батырмасы

Бұл жоба Python және Tkinter кітапханаларын пайдаланып, стектермен жұмыс істеуге арналған қарапайым графикалық интерфейс құруды мақсат етті. Жобада VS Code өңдеу ортасы, git нұсқаларды басқару жүйесі және venv виртуалды ортасы қолданылды.

Бағдарлама қолданушыларға стекке мәліметтер қосу (Push), жою (Pop) және оларды визуализациялау мүмкіндігін ұсынды. Интерфейс екі негізгі терезеден тұрады: бірі сан енгізу және стекте сақтау үшін, ал екіншісі стектің мазмұнын графикалық түрде көрсетуге арналған. Оқиғаларға негізделген бағдарламалау тәсілі арқылы пайдаланушы әрекеттері тиімді өңделді.

Тестілеу барысында интерфейстің негізгі функционалдары мен пайдаланушы ыңғайлылығы жан-жақты тексерілді. Қолданушы енгізу өрісіне тек сандарды енгізе алды, Push және Pop операциялары дұрыс жұмыс істеді, ал визуализация деректердің айқын көрінісін қамтамасыз етті. Сонымен қатар, қателерді өңдеу және үлкен көлемдегі мәліметтерді көрсету қабілеті тексерілді.

Жоба қарапайым әрі түсінікті интерфейс арқылы стектермен жұмыс істеудің негіздерін меңгеруге көмектеседі. Бұл деректер құрылымдарын оқыту және шағын бағдарламалар құру үшін тиімді құрал болып табылады. Бағдарламаның тұрақтылығы мен интуитивтілігі оның негізгі артықшылығы ретінде ерекшеленді.

# Қорытынды

Курстық жобаның негізгі мақсаты стек деректер құрылымын визуализациялау арқылы түсіндіру және стектің жұмысын пайдаланушыға тиімді түрде көрсету болатын. Жобада Python бағдарламалау тілінде Tkinter кітапханасы көмегімен интерактивті визуализатор әзірленді. Бұл визуализатор стекке жаңа элементтер қосу (push), алып тастау (pop) және тексеру (peek) операцияларын көрнекі түрде көрсетуге мүмкіндік береді.

Стек деректер құрылымы LIFO (Last In, First Out) принципіне негізделген және оның қарапайымдылығы мен тиімділігі оны көптеген бағдарламалау тілдері мен алгоритмдерде кеңінен қолдануға мүмкіндік береді. Стек құрылымын түсіну бағдарламалау және алгоритмдерді оқытуда маңызды, себебі ол рекурсивті функцияларды басқаруда, веб-браузерлерде тарихты сақтау мен математикалық өрнектерді шешуде қолданылатын негізгі құралдардың бірі.

Жобаның барысында теориялық негіздер зерттеліп, стек құрылымын қолданудың практикалық мысалдары келтірілді. Стек алгоритмдерінің динамикалық визуализациясы арқылы пайдаланушыларға оның жұмыс принциптері нақты және түсінікті түрде ұсынылды. Сонымен қатар, жобада қолданылған Python және Tkinter технологиялары қолданушы интерфейсі мен деректер құрылымдарын тиімді көрсетуге мүмкіндік берді.

Курстық жобаның негізгі міндеті стектің жұмысын визуализациялау арқылы бағдарламалау мен деректер құрылымдарын түсінуді жеңілдету болды. Бұл мақсатқа жету үшін түрлі визуализация әдістері қолданылды, бұл өз кезегінде студенттер мен бағдарламашыларға стектің жұмысын оңай түсінуге көмектесті.

Жобаның нәтижесінде стек деректер құрылымын қолданудың практикалық маңыздылығы мен тиімділігі айқындалып, оның әртүрлі салаларда, соның ішінде веб-браузерлерде, ойын бағдарламалауда, математикалық өрнектерді шешуде және жасанды интеллектте қолданылуы көрсетілді.

Осылайша, курстық жоба стектің жұмыс принциптерін интерактивті визуализациялау арқылы түсіндірудің тиімділігін көрсетті және бағдарламалау мен алгоритмдер саласында стектердің маңыздылығын тереңірек түсінуге көмектесті.

# Пайдаланылған әдебиеттер тізімі

|  |  |
| --- | --- |
|  | S. R. M., Introduction to Algorithms, MIT Press, 1989. |
|  | T. H., Introduction to Algorithms, MIT Press, 2022. |
|  | Sedgewick, Algorithms. Addison-Wesley, Addison-Wesley, 1983. |
|  | M. A. Weiss, Data Structures and Algorithm Analysis in C++, . |
|  | С. б. Узайр, Mastering Visual Studio Code: A Beginner's Guide. |
|  | Р. Сильверман, Git Pocket Guide, O'Reilly Media, 2013. |
|  | A. D. Moore, Python GUI Programming with Tkinter - Second Edition. |
|  | A. D. Moore, Python GUI Programming with Tkinter, Packt Publishing. |
|  | E. Matthes, Python Crash Course, No Starch Press, 2015. |
|  | W. M.A., Data Structures and Algorithm Analysis in C++. |

Қосымша А Программалық код

import tkinter as tk

from tkinter import messagebox

import random

stack = []

root = tk.Tk()

root.title("Енгізу терезесі")

root.geometry("700x700")

root.config(bg="lightblue")

label1 = tk.Label(root, text="Сан енгізіңіз", font=("Times New Roman", 16, "bold italic"))

label1.pack(pady=50)

def open\_second\_window():

second\_window = tk.Toplevel(root)

second\_window.title("Жұмыс істеу терезесі")

second\_window.geometry("700x700")

label = tk.Label(second\_window, text="Жұмыс істеу терезесі")

label.pack(pady=20)

canvas = tk.Canvas(second\_window, width=500, height=400, bg="white")

canvas.pack(pady=10)

def update\_canvas():

canvas.delete("all")

rect\_width = 80

rect\_height = 30

y\_offset = 400

for i, row in enumerate(reversed(stack)):

for j, value in enumerate(row):

x1 = 200

x2 = x1 + rect\_width

y1 = y\_offset - rect\_height

y2 = y\_offset

canvas.create\_rectangle(x1, y1, x2, y2, fill="skyblue", outline="black")

canvas.create\_text((x1 + x2) // 2, (y1 + y2) // 2, text=str(value), font=("Arial", 12))

y\_offset = y1

matrix\_text = "\n".join(str(row) for row in stack)

matrix\_label = tk.Label(second\_window, text="Матрица:\n" + matrix\_text)

matrix\_label.pack(pady=10)

def push():

if stack:

new\_value = random.randint(1, 100)

stack[-1].append(new\_value)

messagebox.showinfo("Успех", f"Қосылған элемент: {new\_value}")

else:

new\_value = random.randint(1, 100)

stack.append([new\_value])

messagebox.showinfo("Успех", f"Создана новая строка с элементом: {new\_value}")

update\_canvas()

def pop():

if stack:

if stack[-1]:

removed\_value = stack[-1].pop()

messagebox.showinfo("Успех", f"Өшірілген элемент: {removed\_value}")

if not stack[-1]:

stack.pop()

else:

messagebox.showerror("Ошибка", "Матрица пуста! Нечего удалять.")

update\_canvas()

button\_push = tk.Button(second\_window, text="Push", command=push, font=("Times New Roman", 14, "bold italic"))

button\_push.pack(pady=10)

button\_pop = tk.Button(second\_window, text="Pop", command=pop, font=("Times New Roman", 14, "bold italic"))

button\_pop.pack(pady=10)

update\_canvas()

entry = tk.Entry(root, font=("Arial", 14))

entry.pack(pady=10)

def add\_row\_to\_stack():

entered\_text = entry.get()

if entered\_text.isdigit():

cols = int(entered\_text)

row = [random.randint(1, 100) for \_ in range(cols)]

stack.append(row)

messagebox.showinfo("Успех", f"Қосылған матрица: {row}")

else:

messagebox.showerror("Қате", "Тек сан енгізіңіз!")

button\_submit = tk.Button(root, text="Қосылатын элементтер", command=add\_row\_to\_stack, font=("Times New Roman", 14, "bold italic"))

button\_submit.pack(pady=10)

button\_visualize = tk.Button(root, text="Визуализация", command=open\_second\_window, font=("Times New Roman", 14, "bold italic"))

button\_visualize.pack(pady=10)

root.mainloop()

