**КЕЗДЕЙСОҚ САНДАР ГЕНЕРАТОРЫМЕН БАҒДАРЛАМАЛАУ ЕРЕКШЕЛІКТЕРІ**

**Студент: 1304000 "Электронды есептеу техникасы және бағдарламалық қамсыздандыру" мамандығы 2-курс, 1221 топ**

**Серікбаев Нұрдәулет Талғатұлы**

**"Ақтөбе ауылшаруашылық колледжі" МКҚК**

**Қазақстан Республикасы**

**АННОТАЦИЯ**

Бұл ғылыми жобада кездейсоқ сандар генераторы, оның мүмкіндіктері мен түрлері туралы ой қамтылған және тұрмыста, ғылым мен техникада кездейсоқ сандар генераторын қолдану ерекшеліктері қарастырылған. Сонымен қатар, Turbo Pascal, С#, С++ бағдарламалау тілдерінде кездейсоқ сандар генераторымен бағдарламалау мүмкіндіктері талданып, Delphi интегралды ортасында кездейсоқ сандармен жұмыс жасайтын қосымша құру үлгісі көрсетілген.

В этом научном проекте указаны генераторы случайных чисел, его возможности, виды и их разнообразие использования в быту, в науках и в технике. А так же указаны возможности программирования генераторов случайных чисел на языке Turbo Pascal, C#, C++ и примеры приложения с случайными числами в интегрированный среде Delphi.

This science project is about random number of generator, their abilities and types. The random number of generator is peculiarities of consided and uses in common life, science and technic. Also the Turbo Pascal, C#, C++ language programme was consideced by random number of generator and additional model random number of generator showed in the middle of Delphi integral

Әлемдегі барлық процесс кездейсоқтықтан немесе алдын ала қарастырылған жоспар не болмаса заңдылық негізінде өтіп жатады. Кездейсоқ оқиғалар - әркез өмірімізде кездесетін жайттардын бірі болып саналады. Ғылымдардың көбісі бұрыннан кездейсоқ оқиғалардың заңдылықтарын зерттеумен айналысып келген болатын. Иә, онысы рас, кездейсоқ оқиғалардың өз заңдылықтары да бар және оларды біліп-тану біз үшін маңызды. Себебі, кездейсоқ оқиғанын заңдылығын біле отырып, лотерея ұтысынан бастап ғылыми тәжірибенің статистикалық өңдеуіне дейін жетуге болады. Нақты алғанда өміріміздің кез - келген саласында кездейсоқ сан қолданамыз.

Кездейсоқ сандар генераторы — кездейсоқ сандар беретін құрылым немесе бағдарлама. Әдетте, кездейсоқ сандар генераторы - бағдарламалау тілдерінде стандартты функциялар мен процедуралар құрамына кіретін бағдарлама.

Кездейсоқ сандардың аппараттық генераторы - өтіп жатқан физикалық процесс параметрлері негзінде кездейсоқ сандар тізбегін генерациялайтын құрылғы. Кездейсоқ сандардың аппараттық генераторы - жай жұмыс жасайды немесе аралас тізбек өндіреді. Мұндай генераторларды қолдану пәндік обылыстың және генератордың өзінің құралдарынан тәуелді. Кездейсоқ сандар генераторын қолдану деңгейі әртүрлі, яғни қарапайым ойыншықтан күрделі шифрлауға дейін. Сәйкесінше генераторға қойылатын талап та күшейеді.

Кілтті генерациялау үшін бізге кездейсоқ сандарды генерациялауыш (random number generator - RNG) керек. Жақсы кездейсоқтықтың генерациясы – расында да көптеген криптографиялық операциялардың ең қажетті және сонымен қатар аса ауыр бөліктерінің бірі.

Біз кездейсоқтықтың өзімен расында нені білдіретінін онша толық қарастырмаймыз. Осы терминнің көптеген әдемі математикалық анықтамалары бар, бірақ олардың бәрі бұл жобада қарастыру үшін өте ауыр. Ал формальсыз емес жағынан егер де қолданушы кездейсоқтықпен күресу үшін белсенді әрекеттер қолданатын болса да, кездейсоқтықты қаскүнем үшін болжамсыз мәліметтер мәндері ретінде анықтауға болады.

Көптеген криптографиялық функцияларға жақсы кездейсоқ сандардың генераторлары керек. Біз А және Б қолданушыларға белгілі кілт бар деп жорамалдадық. Бұл кілт бір жерде шығарылу керек. Кілттерді таңдау үшін кілттерді басқару жүйесі кездейсоқ сандардың генераторын қолданады. Егер генератор онша мықты болмаса, нашар кілт алынады. Тап осы жағдай Netscape шолушысының ерте нұсқаларының бірінде болды.

Кездейсоқтықтың өлшемі энтропия (entropy) деп аталады. Мен бұл жерде осы сұрақтың математикалық бөліктерін келтіріп жатпаймын, жай ғана энтропияның не екенін түсіндіру үшін барлығын жасауға тырысамын. Егер бізде мүлде кездейсоқ түрмен таңдалатын 32-биттік сөз болса, онда оның 32 бит энтропиясы болады. Егер де 32-биттік сөз әрқайсысының шығу ықтималдығы 25%-ға тең тек 4 мән қабылдай алатын болса, онда бұл сөз тек 2 бит энтропияға ие болады. Энтропия мәндегі биттің санын емес, осы мәнде сіздің қаншаға сенімсіз екендігіңізді көрсетеді. Көбірек суреттеу үшін энтропияны қысудың тамаша алгоритімін қолдану кезінде мәнді беру үшін керек биттің орташа саны ретінде қарастыруға болады. Назар аударыңыздар, мәннің энтропиясы осы мән туралы қанша білетініңізге байланысты. 32-биттік кездейсоқ сөздің 32 бит энтропиясы бар. Енді осы сөздің мәні туралы келесілер нақты белгілі деп қарастырайық: сөздің 18 биті 0-ге тең, ал 16 бит – 1-ге тең. Бұндай талаптарға жуықтап 228,8 мән қанағаттандырады, осыдан шығатыны, сөздің энтропиясы тек 28,8 битті құрайды. Басқа сөзбен айтқанда, неғұрлым мән туралы көбірек білсек, соғұрлым оның энтропиясы аз болады.

Кездейсоқ үлестірілуі бір қалыпты болмайтын мәндер үшін энтропияны есептеп шығару біраз қиындау.

Тамаша әлемде «ақиқат кездейсоқ» сандарды қолданған дұрыс болар еді. Өкінішке орай, біздің әлем тамаша емес және онда шындығында кездейсоқ мәліметтер табу өте қиын.

Кәдімгі компьютерлерде энтропияның бірнеше көздері бар. Осындай көздердің мысалы ретінде көбінесе перненің дәл басылу уақыты мен тышқанның орын ауыстыруының дәл уақыты келтіріледі. Кезінде кейбір ғалымдар қатты дискіге жету уақытының оның корпусының ішіндегі турбуленттікпен шақырылған тербелістерінің кездейсоқтығына зертеулер жүргізген. Осы барлық энтропиялардың көздері күмән туғызады, өйткені кейбір нақты жағдайларда қаскүнем кездейсоқтықтың көзінің үстінен есептеулер жүргізуі немесе осы көзге әсер етуі мүмкін.

Кездейсоқ сандардың аппараттық генераторы - өтіп жатқан физикалық процесс параметрлері негзінде кездейсоқ сандар тізбегін генерациялайтын құрылғы. Мұндай құрылғылар жұмысы фотоэлекрлік эффектілер, жылылық шулар секілді кванттық құбылыстардағы энтропияның сенімді көзі ретінде қолдануға негізделген. Кванттық процеске негізделген кездейсоқ сандар генераторы көбіне арнайы күшейткіштен және түрлендіргіштен тұрады. Күшейткіш өтіп жатқан физикалық құбылыс нәтижесінде алынған өте әлсіз сигналдарды күшейтеді. Түрлендіргіш сигналдарды цифрлық түрге түрлендіреді.

Ақиқат кездейсоқ сандарды біз тек жалғанкездейсоқ сандардың генераторларына кіріс беретін бастапқы сандарды табу үшін ғана қолданамыз. Бізде бастапқы сан табылғаннан кейін генератор өмірге кездейсоқ (толығырақ жалғанкездейсоқ) сандардың кез-келген санын шығара алады. Керек болған жағдайда біз ақиқат кездейсоқ сандарды кездейсоқ сандардың генераторының бастапқы санына қоса аламыз. Бұл, егер қаскүнем қандай да бір түрмен бастапқы санды біліп алған болса да, ақырғының шығыс мәліметтері ешқашан толық болжамды болмайтындығына кепілдік бере алады.

Ақиқат кездейсоқ сандар жалғанкездейсоқ сандардан гөрі жақсырақ деген теориялық ой бар. Кейбір криптографиялық хаттамалар үшін ақиқат кездейсоқ сандарды қолдану кезінде шабуылдың белгілі кейбір түрлері мүмкін болмайтындығын дәлелдеуге болады. Мұндай хаттама сөзсіз қорғалған (unconditionally secure) деп аталады. Егер де жалғанкездейсоқ сандардың генераторларын қолданатын болсақ, хаттама қаскүнем бұл генераторды бұза алмайды деген шарт болған жағдайда ғана қауіпсіз болады. Мұндай хаттама есептеулер бойынша қорғалған (computationally secure) деп аталады. Бұл айырмашылықтың толық білетін теоретиктер үшін ғана мағынасы болады. Барлық криптографиялық хаттамалар көбіне арқашан есептеуіш жақындалуға негізделген. Бұндай жақындалудың нақты бір шабуыл түріне байланысты шығарылуы жақсы жақсаруға әкелмейді. Сонымен қатар, сөзсіз сақталуды қамтамасыз етуге қажетті ақиқат кездейсоқ сандардың шығаруы соншалықты қиын, мұндай сандарды қолдану әрекеті жүйенің қауіпсіздігін тек бұзуы мүмкін. Ақиқат кездейсоқ сандардың генераторының кез-келген нашар орны қауіпсіздіктің жоғалуына әкеліп соғады. Басқа жағынан қарағанда, егер ақиқат кездейсоқ сандарды жалғанкездейсоқ сандардың генераторына бастапқы санды табу үшін ғана қолдансақ, онда өзімізге әр түрлі энтропиялардың көздерін қолдануды жасай алу, сөзсіз, біздің расында да қауіпсіз жүйені құруымыздың мүмкіндіктерін көтереді.

Генератор белгіленген ұзындықтың кейбір жағдайын кез келген ұзындықтың шығыс мәліметтеріне ауыстырады. Өзінің мәні бойынша генератор – бұл тек қана есептеуіш жағдайында жұмыс жасайтын блоктық шифр. Еске салайық, есептеуіш жағдайы немесе CTR мәліметтердің біз шығыстар ретінде қолданатын кездейсоқ ағынын генерациялайды. Егер қолданушы немесе қосымша кездейсоқ мәліметтер сұрайтын болса, генератор өзінің алгоритмін қосып жалғанкездейсоқ мәліметтер шығарады. Енді қаскүнем генератордың күйін кезекті сұраныстан кейін алуға мүмкіндігі болсын деп есептейік. Егер генератормен берілген алдыңғы нәтижелер дескридитацияланбаса жақсы болар еді. Ол үшін әрбір сұраныстан кейін тағы 256 бит жалғанкездейсоқ мәліметтерді генерациялаймыз және оларды шифрлаудың жаңа кілті ретінде қолданамыз. Ескі кілт осы кезде алдыңғы сұраныстар туралы ақпараттың шығуын болдыртпау үшін жойылады.

Генерацияланған мәліметтер статистикалық кездейсоқтыққа ие болуы үшін бір мезгілде аса көп мәліметтерді генерациялап керегі жоқ. Расында да, ақиқат кездейсоқ мәліметтерде блоктың мәндері қайталануы мүмкін, ал есептеуіш режиміндегі шығыс мәліметтері ешқашан қайталанатын блоктарды шығармайды. Бұл мәселені шешудің бірнеше жолдары бар. Мысалы, практика жағынан блогтың ақиқат кездейсоқ кезегінен статистикалық ауытқуын болдырмайтын тексті шифрлаудың блогының тек бір жартысын қолдануға болады. Блогтық шифрлаудың орнына екінің бірі ретінде жалғанкездейсоқ функцияны (pseudorandom function) қолдануға болады, бірақ бізге әлі жақсы зерттелген және тиімді ұсыныстар кездескен жоқ. Осы жағдайда жасауға болатындардың ең оңайы – ол бір сұранысқа берілетін кездейсоқ мәліметтердің байт санын шектеу. Бұл ақиқат кездейсоқ кезектіктен статистикалық ауытқуын қиындатады.

Генератор өзі өте пайдалы модуль болып табылады. Ол көптеген орындауларда тек Fortuna ЖКСГ (жалған кездейсоқ сандар генераторы) компоненті ретінде ғана емес, сонымен қатар оның интерфейсінің бөлігі ретінде қолданылуы мүмкін деп есептейміз. Мысалға Монте-Карло әдісімен модельдеуді орындайтын бағдарламаны алсақ. Біз бұл моделдеу кездейсоқ болуын қалаймыз делік. Онымен қоса, біз керек болған кезде керекті есептеулерді (мысалы, отладка немесе тексеру үшін) жасауға болатындай болуы керек. Ол үшін бағдарлама басында бір рет операциялық жүйенің орнатылған кездейсоқ сандар генераторын шақыруға болады. Оның көмегімен біз кездейсоқ бастапқы санды аламыз. Бұл сан бағдарламаның шығыс мәліметтерінің бөлігі ретінде алынуы мүмкін, одан кейін біздің генератормен моделдеуге керекті қалған кездейсоқ мәліметтерді есептеуге қолданылуы мүмкін. Генератордың шығыс бастапқы санын біле отырып барлық есептеулердің дұрыстығын бақылап отыруға болады. Ол үшін бағдарламаны тағы да сол кіріс мәліметтерімен және бастапқы сандарымен жіберу жеткілікті.

Кездейсоқ сандардың аппараттық генераторы - жай жұмыс жасайды немесе аралас тізбек өндіреді. Мұндай генераторларды қолдану пәндік обылыстың және генератордың өзінің құралдарынан тәуелді.

Кездейсоқ сандар генераторын қолдану деңгейі әртүрлі, яғни қарапайым ойыншықтан күрделі шифрлауға дейін. Сәйкесінше генераторға қойылатын талап та күшейеді. Генераторлар сапасын бағалауға арналған арнайы тесттер бар. Оның негізгілері:

- Жиілік тест - биттер тізбегі мен тізбегінен тұрады. және саны шамамен тең болуы керек.

- Бірдей биттер тізбегіне тест 000...0 немесе 111...1 түріндегі бірдей биттер қатары ізделінеді. Ұзындығынан тәуелді кездейсоқ қатарлар орналасу жиілігі ақиқат кездейсоқ сандар орналасуына сәйкес келуі керек.

- Спектрлік тест - шыққан тізбекке Фурье үзілісті түрлендіруі қолданылады. Алынған спектр тізбектің периодтық құрылымында айтылғандай маңызы бар бос орын болмауы керек.

- Автокорреляциялық тест- бір - біріне қатысты тізбек көшірмелерінің арасындағы корреляцияны есептейді.

*Excel- де* кездейсоқ сандар генераторы екі тәсілмен жүргізіледі: санды бағдарламалық әдіспен генерациялау немесе стандартты функцияларды қолдану арқылы. Мысалы, біз -10 мен 10 аралығынан кездейсоқ сан алуымыз керек. Біздің мысалда ол мына түрде болады: =СЛУЧМЕЖДУ(-10;10). Осы формуланы қолдансақ, ағымдағы ұяшықтан -10 мен 10 аралығынан кездейсоқ бір санды көре аламыз. екінші бір функция СЛЧИС() - 0-ден үлкен және 1-ден кіші кез келген кездейсоқ санды қайтарады. СЛЧИС() функциясының СЛУЧМЕЖДУ() функциясынан айырмашылығы онда аргументтер болмайды.

*iRandomizer***-** бұл ондаған, жүздеген, тіпті мыңдаған сандардың ішінен рандомизациялау және араластыру арқылы кез - келген бір санды таңдауға мүмкіндік беретін Iphone ортасындағы кең танымал бағдарламалық өнім.

Бұл қосымша өмірлері ойын- сауықпен тікелей байланысты адамдар үшін қажет. Мысал үшін сізге мыңдаған санның ішінен кез- келгенін таңдау керек болсын немесе карта номерін таңдау немесе қандайда бір бәйге жеңімпазын анықтау керек болса, Iphone- ға арналған iRandomizer сізге көмектеседі. Ол осының бәрін санаулы секундтарда жүзеге асырыр, нәтижені экранға шығарады.

Қандай да бір шамаға байланысты бағдарлама нәтижесі кездейсоқ болуы керек жағдайлар туындайды. Осындай мүмкіндіктерді жасау үшін көптеген бағдарламалау тілдерінде кездейсоқ сан беретін стандартты функциялар бар. Негізінен сандар кездейсоқ емес, жалған кездейсоқ болады. Себебі жасанды түрде кездейсоқ сан жасау мүскін емес. Көбіне қандай да бір коэффициент алынып және оның арқасында келесі "кездейсоқ" сан есептеледі.

Паскаль бағдарламалау тілінде берілген диапазон аралығында жалған кездейсоқ сандарды өндіру үшін random функциясы қолданылады. Оны қолданар алдында көбінесе кездейсоқ сан генераторы механизмінің дұрыс жұмыс жасауын тағайындайтын randomize процедурасы қолданылады, әйтпесе бағдарлама үнемі бір ғана нәтижені шығарып беріп тұрады. Randomize процедурасы барлық келесілерді есептейтін тізбектің бастапқы мәнін береді. Бағдарламаның әрбір орындалғаны сайын бұл мән әртүрлі болады, сондықтан да random функциясы нәтижесі де әртүрлі болады. Random функциясы 0 мен 1 аралығында кездейсоқ санды генерациялайды. Егер жақша ішінде аргумент көрсетілсе, онда 0- ден жақша ішінде көрсетілген мәнге дейінгі (мәннің өзін қоспағанда) аралықта кездейсоқ сан генерациялайды. Мысалы, random (10) функциясы [0..10) диапазонындағы кез - келген сан алынады. Егер 0- ден басқа бастапқы мәннен басталатын диапазон аралығынан кездейсоқ санды генерациялау үшін математикалық қулықты жүзеге асыруға тура келеді. Мысалы: -100...100 диапазонынан кездейсоқ сан генерациялау үшін мынадай өрнек жазса жеткілікті: random(200)-100; нәтижесінде алдымен, [0..199] диапазонынан сан алынады, содан кейін сол саннан 100 шегеріледі. Егер кездейсоқ сан 100- ден кіші болса, нәтиже теріс сан болады.

Бағдарламалауда қандай да бір әрекеттерді орындау үшін кездейсоқ санды қолдану қажеттілігі туындайды. Ал кездейсоқ санды алу үшін әрине, өз функцияңды құруға болады немесе стандартты кітапханалық функцияларды қолдануға болады. Сонымен, С++ те кездейсоқ сандарды генерациялау үшін <stdlib.h> кітапханасында сипатталған rand() функциясы қолданылады. Дегенмен бұл бағдарламаны орындау кезінде бір ғана мән қайтарады. Осындай қиындықты шешу мақсатында rand() функциясымен srand() функциясын қолдану керек.

Оның сипатталуы: *void srand (unsigned int);*

Жазылу үлгісі: *void srand (unsigned int seed) {next=seed;}*

Бұл функция next бастапқы санын оған жаңа мән беру арқылы өзгертіп отырады. Функция нәтижесін аргумент ретінде қабылдайды. Негізінен бұл санды үнемі қолмен енгізуге болады. Көбінесе srand() функциясының өлшемі ретінде секундпен берілген жүйелік уақыт беріледі. Бұ бірден - бір тиімді әдіс. Бұл сан әрқашан әртүрлі болады, сәйкесінше rand() функциясының нәтижесінде кездейсоқ сандар аламыз. Енді srand() функциясына ағымдағы жүйелік уақытты беру керектігін ойластыру керек. Ол үшін <time.h> кітапханасында сипатталған time() кітапханалық функциясы қолданылады: *time\_t time(time\_t\*);*

Енді мынадай мәселені қарастырайық. Мысалы, берілген интервалда 0..9 диапазонынан кездейсоқ сан генерациялау керек болса, онда rand() функциясының мынадай синтаксисі бар: m = a + rand() % b; Мұнда а- генерация басталатын бастапқы нүкте. b- қозғалыс өлшемі, генерация жүргізілетін интервалды анықтайды. Біздің жағдайда а=0, b=10. Ал егер 30-дан 50 аралығынан кездейсоқ санды генерациялау үшін функция мына түрде жазылады: *m = 30 + rand() % 21;*

Әлемдегі барлық процесс кездейсоқтықтан немесе алдын ала қарастырылған жоспар не болмаса заңдылық негізінде өтіп жатады. Нақты алғанда өміріміздің кез - келген саласында кездейсоқ сан қолданамыз. Монетаны лақтырғанда, покер немесе лоторея ойнағанда, сандық құпиясөз құрғанда біз кездейсоқ сан қолданамыз немесе арнайы ойластырылған кездейсоқ сан генераторын пайдаланамыз. Кездейсоқ сан генераторы - бұл бір - бірінен тәуелсіз түрде сандарды генерациялайтын алгоритм. кездейсоқ сандар гераторының жұмысы алдын - ала жазылған алгоритмнен тәуелді болады. Осыдан соң жалған кездейсоқ сандар туралы әңгіме айтуға тура келеді. Сондықтан мынадай қорытынды жасауға болады: біздің әлемімізде ештеңе де кездейсоқ емес, ал болса ол тек кездейсоқ сияқты көрінеді. көптеген ғылыми мақалаларға сүйенсек, кездейсоқ сандар генераторы атауын жалған кездейсоқ сандар генераторы деп атау дұрысырақ болады. тек ыңғайлылық үшін "жалған" сөзі айтылмайды. Кездейсоқ сандар генераторларының алгоритмдері нақты жағдайда бағдарламалау тілінен және есептеу платформасынан тәуелді. Барлық генератор түрлерінің жұмыс жасау принципі қарапайым ғана- берілген мәндер аралығынан кездейсоқ бір санды таңдайтын ішкі функция қолданылады. Қазіргі кезде кездейсоқ сандар генераторы көбіне онлайн - покеррумдарда қолданылады. Сол себептен осындай сайттар иелері өз алгоритмдерін және кездейсоқ сандар генераторы сертификатын көрсетуге ұмтылады, яғни көзбояушылықтың жоқ екеніне сендіреді.

Сонымен қатар, осындай алгоритмдерді қолданатын тағы бір қызмет саласы бар, ол- криптография. Жаңа әрі қолжетімсіз құпиясөздерді құру үшін осы тәсіл тиімді. Осындай алгоритмді түсініп, комбинацияны есептеу үшін тәуекел еткенмен де, тіпті, тәжірибелі хакерлер үшін де оның сырын ашу мүмкін болмай отыр.

Көбінесе кездейсоқ сандар генераторы лотореяда ұтыс тіккенде қолданылады. Мүмкін сәйкес келер, ықтималдық өте аз, дегенмен "үмітсіз шайтан" деген бар емес пе?

Кездейсоқтық – заңдылықтың жеке бір жағдайы. Жан-жағыңызға қараңыз, сонда сіз әлемнің барлық бұрышында да кездейсоқтық орын алатынын байқайсыз. Кездейсоқ күйде қар жерге түседі, жолда жүріп бара жатқан машиналар жиынтығы да - кездейсоқ. Сондықтан бағдарламалауда шынайы құбылыстарды моделдеу үшін кездейсоқ сандар қолданылады. Жалпы алғанда барлық бағдарламалау тілдерінде айнымалыға кездейсоқ сан меншіктеу мүмкіндігі бар. Көбіне ол үшін алдымен жедел жадта кездейсоқ сандар тізбегін құрайтын арнайы генераторды іске қосу керек . Содан соң осы тізбектен сан алып, оны айнымалыға меншіктеу мүмкіндігі болады.

С# тілінде кездейсоқ сан генераторы үшін Random класы жұмыс жасайды. Random басқару элементті болып табылмайды, сондықтан бағдарламашы бұл класс объектісін бағдарламаны жасақтау кезеңінде де, бағдарламаны іске қосқанда да көре алмайды. Тағайындау, яғни генераторды іске қосу Random обьектісін құрған кезде жүзеге асады. Генератормен жасалған сандар тізбегінен санды алу үшін осы класта арнайы функциялар анықталған. Осы класты қолдануды және оның функцияларын шақыруды жариялау үшін ″Русское лото ″ ойынын моделдейік.

Ойында фишка - бөшке салынған қалтаны ұстайтын жүргізуші болады. Бөшкелерде 1 -ден 99-ға дейінгі сандар көрсетілген , жүргізуші қалтадан бір фишка- бөшкеден алып отырады. Алынған бөшкеден сан кездейсоқ болады. Біздің жағдайда жүргізуші ролін компьютер орындайды.

|  |  |  |
| --- | --- | --- |
| Не істеу керек? | Қалай істеуге керек? | Код мысалы |
| Фишка - бөшкелерге арналған қалта дайындау | Осы кездейсоқ сандар генераторы болып табылады,яғни Random класының обьектісі | Random myRnd |
| Қалтаға фишка -бөшкелерді салып толтыру | Обьект құру үшін new кілттін | MyRnd=new Random(); |
| Алынған фишка - бөшкені қоятын орынды дайындау | Бүтін типпті айнымалыны хабарлаймыз, себебі бүтін сан генерациялау талап етіледі | Int x |
| Қалтаға қолды салып және кезекті бөшкені алып шығу | Кездейсоқ санды генерациялау және оны х айнымалыны жазамыз | Х=rnd.Next(1,99) |

1- кесте

Нәтижесінде біз new функциясы көмегімен тек қана жаңа генератор құрамыз, ал кезекті кездейсоқ санды Next функциясының көмегімен алатынымызды көреміз.Тип int болғандықтан бұл функция бүтін сан қайтаратын болады.

Random класымен Next функциясының 3 түрлі тәсілімен шақыруға болады:

1. Next() - 0..231-1 - диапазонынан кездейсоқ сан қайтарады.
2. Next(myMax) - 0..myMax диапазонынан кездейсоқ сан қайтарады.
3. Next(myMin, myMax) - myMin..myMax диапазонынан кездейсоқ сан қайтарады.

Үш нұсқада да берілген диапазон аралығынан кездейсоқ сан алатын боламыз. Бірақ осы диапазонды әртүрлі етіп беруге болады: шектеулер қабылдау, үнсіз келісім бойынша ұсынылған немесе өз мәніңізді беру.

Қатаң түрде айтқанда, математикалық алгоритм негізінде кездейсоқ сан алу мүмкін емес. Джон фон Нейман бір сөзінде былай дейді: "Кімде - кім кездейсоқ сан алудың арифметикалық әдісіне әлсіздік танытса, ешқандай күмәнсіз күнәлі болады." Компьютер жалған кездейсоқ деп аталатын сандарды генерациялайды, бірақ олар кездейсоқтықтың кейбір статистикалық өлшемдеріне сай келеді. кездейсоқ сандар мен жалған кездейсоқ сандар арасында бір ғана айырмашылық бар, яғни жалған кездейсоқ сандар тізбегінің шегі болады және ондағы сандар көршілерімен өзара байланыста болады.

Жалған кездейсоқ сандарды генерациялау бастапқы мәннен басталады. Осы бастапқы мәндермен арифметикалық амалдар тізбегін жасайды, осы амалдарды жалғастыра отырып, екінші сан және т.с.с. алатын тізбектің бірінші саны алынады. Бірдей бастапқы мән қолданған жағдайда сандардың бірдей тізбегі құрылады. Әртүрлі тізбек алудың бірден - бір жолы - бастапқы мәннің жүйелік уақыттан тәуелді болуы. тізбекті генерациялауға арналаған бастапқы мән генератор құру кезінде компьютерге белгілі болуы керек. С# - та генератор myRnd = new Random(); жолы арқылы құрылады. New кілттік сөзі конструктор деп аталатын функцияны шақырады. Объект құратын барлық класта бірнеше констуктор болуы мүмкін. Random класында конструктор артық жүкті болады: объект құрудың екі әдісі бар.

Біріншісі - үнсіз келісім бойынша конструктор, параметрсіз- бос жақшалар шақырылғаннан кейін: Random myRnd = new Random();

Екіншісі - жақшада көрсетілген бүтін параметрлермен берілген конструктор:

int n = 5;

Random myRnd = new Random(n);

Қандай бастапқы мәнді алу керек екенін конструктор көмегімен бағдарламашы өзі таңдайды. Егер параметрлі конструктор қолданылса, онда бұл сан бастапқы мәнді есептеуге арналады. Біздің жағдайда ол 5 болады. Сондықтан бағдарламаны қайта іске қосқанда дәл осы тізбекті қайта алатын боламыз. Үнсіз келісім бойынша алынған конструкторды қолдансақ, ағымдағы жүйелік уақыт көмегімен әртүрлі тізбек алатын боламыз.

**ҚОЛДАНЫЛҒАН ӘДЕБИЕТТЕР**

1. «Delphi 6», В.Гофман, А. Хомоненко. bhv, Санкт-Петербург. 2001 ж.
2. «Delphi7». А.Хоманенко, В.Гофман, Е.Мещеряков, В.Никифоров – Санкт-Петербург БХВ- Петербург, 2004 г.
3. «Delphi7 Справочное пособие» А.Я.Архангельский М, 2004 г.
4. «Бағдарламалау тілін оқыту негіздері» А.Хамметов, М. Ғалымжанова,

В.Махатова "Foliant" баспасы, Астана - 2011

1. «TURBO Pasсal» Ж.Қ.Масанов, Б.А.Бельгибаев, А.С.Бижанова,

Қ.Қ. Мақұлов Алматы 2004

1. «Обьектілі бағытты бағдарламалау негіздері» А.Аймұқатов. "Foliant" баспасы, Астана - 2011
2. «С/C++ в задачах и примерах» Санкт-Петербург Н.Культин 2005ж.
3. «С++ обьектно- ориентированное программирование» практикум Т.А. Павловская, Ю. А. Шурпак
4. «С/C++ Программирование на языке высокого уровня»Т.А.Павловская
5. «Полный справочник по С #» Г. Шилдт