|  |  |
| --- | --- |
| 1 | Алгоритмдердің түсінігі мен қасиеттері, деректер құрылымына түсініктеме беріңіз. Мысал келтііңіз |

Алгоритм, алгорифм (ағылш.: algorіthm, algorіsmus — Әл-Хорезмидің атынан шыққан) — бастапқы берілген мәліметтермен бір мәнде анықталатын нәтиже алу үшін қай амалды (жұмысты) қандай ретпен орындау қажеттігін белгілейтін есептерді (мәселелерді) шешу (математикалық есеп-қисаптар орындау, техникалық объектілерді жобалау, ғылыми-зерттеу жұмысын жүргізу т.б.) тәсілдерінің дәл сипаттамасы.

Жалпы Алгоритм деп алдын ала не істеу керек екені дәл көрсетілген есептеу үрдісін айтады. Есептеу үрдісі қандай болса да алғашқы мәндерден бастап, сол арқылы толық анықталған қорытынды шыққанша жүргізіледі. Алгоритм ұғымының алғышартына алгоритмдік үрдіспен қатар мүмкін болатын алғашқы деректер жиынтығының нұсқауы және қорытынды алуға байланысты жүргізілген үрдістің аяқталғандығын көрсететін ереже енеді. Белгілі бір бастапқы деректердің жиынына қолданылған Алгоритм тиянақты қорытындыға келмеуі немесе есептеу барысы аяқталмай тоқталуы мүмкін. Егер есептеу үрдісі белгілі бір қорытынды алумен аяқталса (не аяқталмай қалса), онда Алгоритм мүмкін болатын бастапқы деректерге қолданылады (не қолдануға болмайды) деп ұйғарылады.

Мысалы есыкты ашу ушын быз:

* Кылтты калтадан шыгарамыз
* Кылтты есыкке тыгамыз
* Кылтты 2 рет бураймыз
* Кылтты есыктен шыгарамыз

Алгоритмнің бұйрықтары бірінен кейін бірі кезекпен орындалады. Бағдарлама Алгоритм тілінде жазу, бейнелеу мағынасын береді. Компьютерде Алгоритмнің сызықты, тармақты, циклді, логикалық, модельдік, параллельдік, тізбекті т.б. түрлері қолданылады. Быз колданган мысал сызыктык алгоритмге жатады

|  |  |
| --- | --- |
| 2 | Абстрактілі есептеу машииналары, Тюринг және Пост машинасына айырмашылықтары туралы жазыңыз. |

Алгоритмге бірнеше анықтамалар берілді, олардың ішіндегі ең алғашқысы ағылшын математигі А.Тьюрингтікі болды. Ол 1936 жылы бірнеше абстрактілі машинаның үлгісін көрсетті және осы машинаның жұмыс үлгісінің көмегімен ереженің орындалу әрекетін қалыптастырды.

Тьюринг машинасы іс-тәжірибе жүзінде қолдануға болмайтын абстракция болып табылады. Сондықтан Тьюринг машинасы алгоритмдері басқа амалдармен орындалуы тиіс. Тьюринг машинасының қолданысында негізгі салдар алгоритм қалыптастыру әртүрлі алгоритмнің тапсырмаларының шешуінің болуын немесе болмауын дәлелдейтін мүмкіндік туғызады.

Тьюринг машинасы алгоритмдерді және дәлелдеуде композициялы алгоритмнің шындықта мүмкін болатынын Тьюринг әр түрлі дәлелді ұсыныстарын көрсетті, келесі тезиспен көрсетуге жағдай жасады: «Кез келген алгоритм сәйкес Тьюринг машинасымен орындалуы мүмкін».

Тьюринг машинасы тезисі дәлелденбейді. Өйткені оның құрылымында «кез келген алгоритм» ұғымы анықталмаған. Әр түрлі алгоритмдерді Тьюринг машинасы түрінде көрсетіп, дәлелдеуге болады. Бірақ класс функциясы осы машинамен есептелінген рекурсиялы функциямен сәйкестелініп дәлелденді.

Тьюринг машинасының формальді емес анықтамасы

Тьюринг машинасы автомат сияқты, екі жақты шексіз лентасы, басы (головкасы) және басқару құрылғысы бар. Басқару құрылғысы бір қалыпты болуы мүмкін, соңғы көпмүше жасайды Q = {q0, q1, ..., qn}. Көпмүше Q Тьюринг машинасының ішкі алфавиті деп аталады.

Тьюринг машинасының есептегіш машинадан принципті айырмашылығы мынада, оның еске сақтау құрылғысы шексіз лентадан тұрады, сондықтан ол шындыққа жанаспайтын қиялдағы машина болып қала береді. Лента ұяшықтарға бөлінген, әрқайсысына алфавиттің бір символы жазылған соңғы алфавит A={a0,a1,...,am}болады, ол Тьюринг машинасының енгізу алфавиті аталады. Тьюринг машинасы функциясын қолдану кезінде ұяшықтардың соңғы саны толтырылмаған болуы мүмкін. Есептегіш (считывающая) басы әр уақытта лентаның ұяшығына шолу, осы ұяшықтағы символға тәуелді және басқару құрылғысының қалпы ұяшыққа жаңа символ жазады немесе оны өзгертусіз қалдырады, ұяшық оңға немесе солға жылжиды немесе орнында қалады. Осыдан барып басқару құрылғысы жаңа күйге ауысады немесе бұрынғысында қалады. Бастапқы құрылғысының қалпы ортасында бастапқы қалып q0 және соңғы қалып qz болып бөлінеді.

Тьюринг машинасының бір тактісі бойынша символды санауға болады, оның орнына жаңасын жазып немесе оны өзгеріссіз қалдырамыз және басын бір ұяшық солға немесе оңға жылжытамыз немесе орнында қалдырамыз.

**Пост абстракты машинасы**, жазатын немесе оқитын түбіртек арқылы не ен жазылып, не ен оқылатын жеке секцияларға (ұяшықтарға) бөлінген ақырсыз таспа болып табылады.

Пост алгоритмдік машинасы алгоритм ұғымын дәлелдеуші

Бұл машинаның Тьюрингтен айырмашылығы – ол өзінің теориясында «машина» емес «алгоритмдік жүйе» деген терминді қолданған.

Оның абстрактылы машинасы бірнеше бірдей секцияларға бөлінген, оқу-жазу инесі бар шексіз лентадан тұрады. Әр секция бос немесе толтырылған болуы мүмкін. Лентаға түк жазылмаса секция бос, лентаға жазылып белгі түссе секция толық деп есептеледі.

Лента жағдайы процесс уақытында өзгермелі болды. Осы лента жағдайы мен оқу-жазу инесінің орны туралы ақпарат Пост машинасының жағдайын айқындайды.

Инені « », метка-белгі М болсын. Секция бос болса, ешбір белгі түспейді. Бір қадам жасағанда ине оңға немесе солға 1 қадам жылжып белгіні салады немесе өшіреді. Программадағы командаларға сәйкес машина 1 жағдайдан келесі жағдайға көшіп отырады.

Әрбір команданың структурасы ХКУ болсын,

Х – орындалатын команда нөмірі,

К – орындалатын әрекет туралы нұсқау,

У – келесі команда нөмірі.

Тьюринг машинасы

Бұл елестегі машина - яғни ―қағаз бетіндегі машина немесе машинаның

математикалық моделі.

Тьюринг машинасы - таза абстракция және ешқашан жасалмаған. Оның

пайдасы тҥрлі есептер шешімінің алгоритмі бар немесе жоқ екендігін

дәлелдеуге болады. Машина белгілі бір алгоритмді орындайтын болғандықтан,

бұл машинаға алгоритмнің қасиеттерінен талаптар қойылады. Біріншіден,

машина толықтай детерминенделген (есептеулер нақты және жалпы тсінікті)

болуы қажет және тапсырылған ережелер жҥйесі негізінде әрекет етуі керек.

Екіншіден, ―бастапқы мәліметтерді енгізуге мҥмкіндік беруі қажет.

Үшіншіден, берілген машинаның жұмыс жасау ережелерінің жҥйесі және

шешілетін есептердің класы машина жұмысы нәтижесін оқи алатындай болып

келістірілуі керек

Тьюринг тезисі кез-келген алгоритмді Тьюринг машинасына салып

шешуге болатынға негізделген.

Алгоритм абстрактілі машина іспеттес.

Тьюринг машинасы – белгілі бір есептерді шығаруға арналған қатаң математикалық құрылым, математикалық аппарат. Бұл аппарат машина деп аталу себебі оның құрамдас бөлігінің және функцияларының есептеу техникасына ұқсауында. Тьюринг машинасының есептеу техникасынан ерекшелігі оның еске сақтау құрылғысы шексіз лентадан тұруында, ал есептеу техникасының еске сақтау құрылғысы қаншалықты үлкен көлемді болса да шектеулі. Сондықтан Тьюринг машинасын лентасы шексіз болғандықтан есептеу техникасы түрінде қолдануға болмайды.

Тьюринг машинасымен жұмыс істеу үшін объектілер туралы ұғымдарға тоқталу қажет.

Әлдебір алфавиттен алынған әріптердің кез келген тізбегі осы алфавитте сөз деп аталады.

Сөздегі әріптердің саны сөз ұзындығы деп аталады. Әріптері жоқ сөзді бос сөз дейді. Олар « » немесе деп белгіленеді.

Әлемдегі барлық объектілерді әртүрлі алфавиттегі сөздер түрінде қарастыруға болады. Сондықтан алгоритмнің жұмыс істеу объектілері сөздер болып табылады.

Алгоритм қолданылатын сөзді енгізілетін сөз дейді. Алгоритмнің нәтижесі шығарылатын сөз деп аталады. Алгоритм қолданылатын сөздердің жиыны алгоритмнің қолданылу облысы деп аталады.

Әрбір Тьюринг машинасында 2 бөлік бар:

1. Ұяшықтарға бөлінген екі жағынан да шексіз лента

2. Автомат - жазу/оқу инесі

Тьюринг тезисі: Кез келген алгоритм үшін сәйкес Тьюринг машинасын құруға болады.

Анықтама. Тьюринг машинасы деп жүйесін айтады. Мұндағы: А – шекті -жиын, Тьюринг машинасының алфавиті, - А алфавитінің бос сөзі, Q –Тьюринг машинасының жағдайын білдіретін шекті жиынның элементі, q1- ТМ-ның бастапқы жағдайы, q0- ТМ-ның тоқтау жағдайы, пассивті жағдай, Т-ТМ-ның жылжу жиыны, - ТМ-ның программасы.

Алгоритм (Тьюринг бойынша) – қойылған есепті шешуге келтірілетін Тьюринг машинасына құрылған программа.

Тьюринг машинасы мен тезисі болашақта да қолданылады, себебі кез келген проблема шешілмейді деп айтуға болмайды, шешілмейтін есеп болмауы керек, оны қандай да бір шешілетін түрге келтіру керек, соған орайластырылған алгоритм құрастыру керек.

|  |  |
| --- | --- |
| 3 | Алгоритмдерді әзірлеу. Сызықтық алгоритмдердің тармақталатын алгоритмдердің, циклдік, алгоритмдерддің сызбаларын құрастыру. |

**Алгоритм** сөзі **9** ғасырда өмір сүрген көрнекті математик Әл-Хорезмидің латын тілінде бұрмаланған аты. Ол бірінші рет арифметикалық амалдарды орындаудың ережелерін тұжырымдаған ғалым.

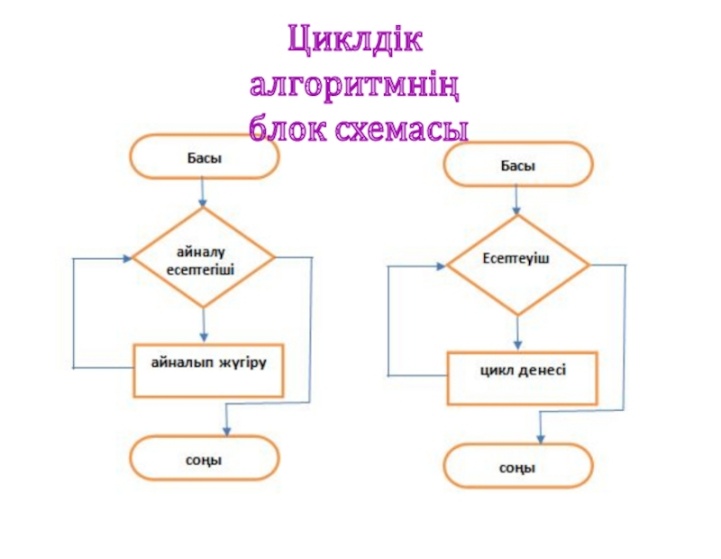
**Алгоритм** дегеніміз – алға қойылған мақсатқа жету үшін немесе берілген есепті шешу үшін түсінікті де нақты ережелер бойынша орындаушыға жинақы түрде берілген реттелген нүсқаулар тізбегі.

**Сызықтық алгоритмдер.** Сызықтық алгоритм қарапайым командалар тізбегінен тұрады. Әрекеттердің тізбектей орындалуын сипаттайтын алгоритм – сызықтық алгоритм деп аталады.

**Тармақталу алгоритмі.** Күнделікті өмірде алгоритмдер негізінен тармақтарға бөлінеді. Бұл тармақтардың орындалуы шарттың орындалуына байланысты болады. Тармақталу алгоритмінде біріншіден, арифметикалық теңсіздік түрінде берілген

Логикалық шарт тексеріледі. Егер шарт орындалса, алгоритм бір тармақ бойынша (1 – ші серия), ал егер орындалмаса, екінші тармақ бойынша (2 – ші серия) орындалады.

**Циклдік алгоритмдер.** Көптеген алгоритмдерде белгілі бір әрекеттер бірнеше рет орындалады. Математикада, есеп шығару барысында айнымалы мәні өзгеруіне байланысты бір теңдеудің көмегімен бірнеше рет есептеледі. Алгоритмнің белгілі бір бөлігі бірнеше қайталанатын болса, ондай процессті

цикл деп атайды. Қайталанатын бөлігі бар алгоритмдер циклдік алгоритм деп аталады. Циклдік алгоритмдер қолдану арқылы программаны қысқа да нұсқа жазуға болады. Қайталану бөлігіне қайталану (циклдік) командалары қолданады. Қайталану саны белгілі немесе белгісіздігіне байланысты циклдер екі түрге бөлінеді. Қайталану саны белгілі болса, ондай цикл арифметикалық, ал егер белгісіз болса, итерациялық деп аталады.

|  |  |
| --- | --- |
| 4 | Алгоритмді жазу тіліндегі қарапайым операцияларға түсініктеме беріңіз. Тізбектелген және тармақталу конструкциясын сипаттап беріңіз. |

Тармақталған алгоритм- орындалуы керек қадам белгілі бір шарттың орындалу немесе орындалмауына байланысты жүзеге асырылатын алгоритм.

Тармақталған алгоритмде «егер», «онда», «әйтпесе» түйінді сөздері қолданылады.  
Бұл алгоритмде міндетті түрде шарт қолданылады.

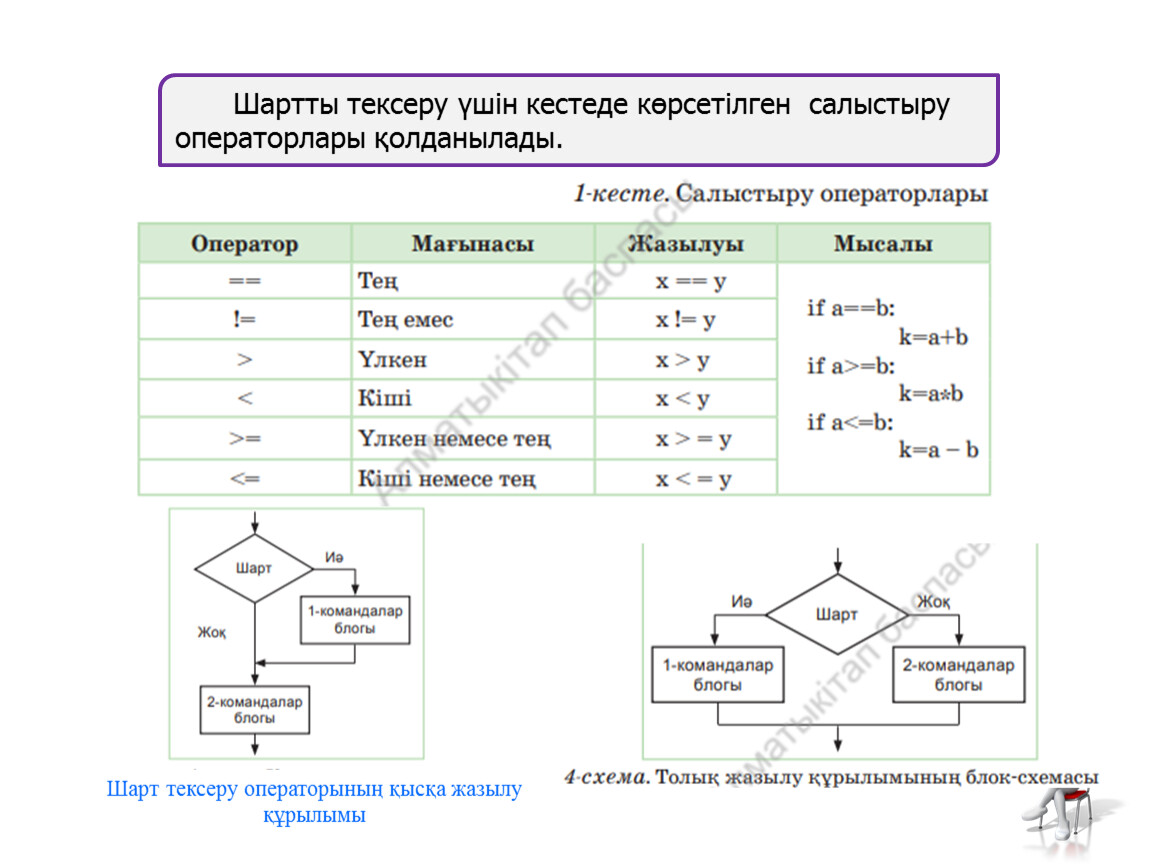
Шарт дегеніміз- тексеру нәтижесінде «иә» (ақиқат) немесе «жоқ» (жалған) логикалық мәндерін қабылдайтын логикалық өрнек немесе пікір.

Шарт дегеніміз не?

Python бағдарламалау тілінде шарт тексеруге арнайы if- else операторы қолданылады. Ағылшынның «if» сөзі «егер», ал «else» сөзі «әйтпесе» мағынасын береді.  
Егер шарт орындалса, яғни ақиқат болса, онда көрсетілген команданы орындауға көшеді.

Шартты тексеру үшін кестеде көрсетілген салыстыру операторлары қолданылады.

Шарт тексеру операторының қысқа жазылу құрылымы



**Тізбектелу алгоритмі**

range

Әрекеттерді қайталау үшін циклдар жазылады.

Цикл ішіндегі операторлар (қайталау керек) шегініспен жазады.

4 рет Hello басып шығару:

for x in range(4):

print('Hello')

# Output:

# Hello

# Hello

# Hello

# Hello

for айнымалы in реттілік: - бұл Python операторы, ол тізбектен 1 элементті сұрыптайды және оны айнымалыға жазады.

x-айнымалы атау. Кез келген болуы мүмкін.

range-python функциясы. Ол бүтін сандар тізбегін қайтарады.

Қос нүктені қоюды ұмытпаңыз :

for x in range(4):

print(x)

# Output:

# 0

# 1

# 2

# 3

range (4) 0, 1, 2, 3 тізбегін қайтарады. Яғни, 4 бүтін сан, 0-ден (оның ішінде) 4-ке дейін (қоспағанда).

Бұл сандарды 1 жолға басып шығару үшін біз әр саннан кейін бос орын қоямыз. Басып шығару функциясында end=' ' (соңында бос орын қойыңыз, әдеттегідей жаңа жолдың белгісі емес).

for X in range (4): # x айнымалы кезек 0 1 2 3

басып шығару (x, end=' ') # x мәнін және бос орынды басып шығарыңыз

басып шығару () # циклден кейін 1 рет \N басып шығарыңыз (жаңа жол таңбасы)

# Output:

# 0 1 2 3

Range(3, 10) функциясы 3-тен (оның ішінде) 10-ға дейін (оның ішінде)сандар тізбегін қайтарады:

for X in range (3, 10): # айнымалы x кезекпен 3 4 5 6 7 8 9

басып шығару (x, end=' ') # x мәнін және бос орынды басып шығарыңыз

print() # печатаем символы жаңа жолдан \n

# Output:

# 3 4 5 6 7 8 9

Range функциясының үшінші дәлелі-санды қанша көбейту керек: Range функциясы(3, 10, 2) әр уақытта 3-тен 10-ға дейін (қоспағанда) +2 сандар тізбегін қайтарады:

for X in range (3, 10, 2): # айнымалы x кезекпен 3 5 7 9

басып шығару (x, end=' ') # x мәнін және бос орынды басып шығарыңыз

print() # печатаем символы жаңа жолдан \n

# Output:

# 3 5 7 9

|  |  |
| --- | --- |
| 5 | Сызықтық алгоритмдердің, тармақталатын алгоритмдердің сызбаларын құрастыру. Мысал келтіріңіз. |

Сызықтық алгоритм командалары осында көрсетілген рет бойынша орындалатын тізбектеле орналасқан командалардан (блоктардан) тұрады.

Амалдардың бұлай бірінен соң бірі реттеліп орындалу тәртібін табиғи атқарылу дейді. Мысалы, төменде көрсетілген Z функциясының сандық мәнін есептеп шығару алгоритмін жасау керек болсын

*Z=aх2+b+cos(aх2+b)-tg(aх2+b)*  
Бұл функцияның мәнін табу үшін алдымен жақшада тұрған aх2+b көпмүшелігін жеке есептеп алу қажет, себебі ол тізбек үш [рет есептеліп](https://kzref.org/1-idista-1mole-sutegi-gazi-bar-idistafi-sutegi-molekulalarini-v2.html), орындаушы машина оған уақытты көп кетіреді. Есептеліп болған Z функциясының мәні қағазға не экранға басылып шығуы тиіс. Жалпы компьютер жадына a, b, х мәндері алдын – ала енгізілуі керек.  
  
**алг**Z функциясын есептеу

(**нақ** a, b, x, z)

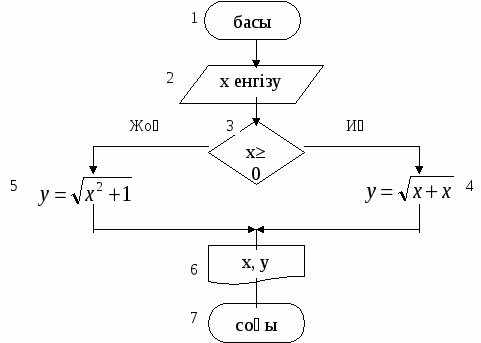
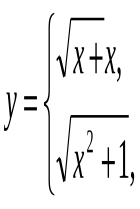
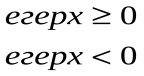
**арг** a, b, x

**нәт** z  
  
**басы**

a, b, x енгізу

t:= aх2+b

z:=t+cost-tgt

**2. Тармақталу алгоритмдері.**Тармақталу алгоритмінде көбінесе арифметикалық теңсіздік түрінде берілген логикалық шарт тексеріледі. Егер орындалса, онда алгоритм бір тармақпен жүзеге асырылады да, соңында екі тармақ қайта бірігеді. Мұндай алгоритмде шартты тексеру ***тармақталу командасы*** деп аталады. Оны алгоритмдік [тілде өрнектегенде](https://kzref.org/2015-jilfi-atar-nauriz-kezeinde-operaciyali-jospardi-iske-asir.html)**егер, онда, әйтпесе, бітті** түйінді сөздері пайдаланылады. Орындалу тәсіліне байланысты тармақталу командасы *«таңдау» (толымды)* және *«аттап өту» (толымсыз)* болып екі түрге бөлінеді.  
**Мысалы:**  

|  |  |
| --- | --- |
| 6 | Деректер түрлері мен құрылымдары. Арифметикалық амалдарға мысал келтіріңіз. Деректер құрылымы ұғымы мен алгоритм арасындағы байланысты түсіндіріңіз |

**Деректер үлгісі туралы түсінік**

**Деректер үлгісі**

Деректер үлгісі - бұл өз еркімен тақырыпты модельдеу құралы.

**Деректер үлгісі** - Бұл мәліметтер базасындағы деректер құрылымын, олардағы деректер құрылымын, сонымен қатар, деректердің рұқсат етілген сілтемелері мен мәндерін, олардың өзгеру реті, оларды өзгертудің реттілігі жиынтығы жиынтығы. Сонымен, деректер үлгісі үш бөлімнен тұрады:

1. Деректер құрылымдық типтерінің жиынтығы.

Мұнда сіз бағдарламалау тілдерімен аналогты жүргізе аласыз, онда алдын ала анықталған деректер құрылымдары, мысалы, скалярлар, векторлар, массивтер, массивтер, құрылымдар (мысалы, түрі) *құрылым* in s) және т.б.

1. Осы құрылымдардың кез-келген бөліктеріндегі ақпаратты табу, шығару немесе түрлендіру үшін осы құрылымдардың кез-келген бөліктерінде көрсетілген деректер түрлеріне және шығыс ережелерінің жиынтығы.

Мұндай операциялар: Деректер құрылымдарын құру және өзгерту, жаңа мәліметтер жасау, бар деректерді жою, жою және өзгерту, әр түрлі жағдайлар туралы мәліметтерді іздеңіз.

Деректер базасының тұрақты күйлерінің жиынтығын және / немесе оның көптеген өзгерістерінің жиынтығын тікелей немесе жанама түрде анықтайтын жалпы тұтастықтың жиынтығы.

Дербес программаны құру үшін программалау тілін білу ғана жеткіліксіз. Программаның түпкі негізі алгоритм ұғымынан құралады. Себебі алгоритм көмегімен программист өзі құрмақшы болып отырған программаға сәйкес мақсатқа жетуі үшін орындауы қажет әрекеттердің тізбегін құрастыруы керек. Алгоритмнің негізгі қызметі – берілген ақпаратты өңдеу арқылы басқа, жаңа ақпарат құру.

Сонымен **алгоритм** дегеніміз белгілі бір мәселені шешу үшін қойылатын мақсатқа бағытталған іс-әрекеттердің тізбегі. Олар **тұрмыстық, есептеу, рекурсивті, қосалқы**деп бөлінеді.Сөз түріндегі әрекеттер тізбегін күнделікті өмірде ешбір роботтың, техниканың көмегінсіз адам өздігінен орындаса ондай алгоритмдерді **тұрмыстық алгоритм** дейді. Мысалы: дүкенге барып азық-түлік әкелу, [сабаққа дайындалу](https://melimde.com/sabaa-dajindalu-tapsirmalari-1-tairip-memleketti-arji-izmeti-j.html), т.б.Формула көмегімен шығарылатын, есептеуді қажет ететін, күрделілігіне байланысты белгілі бір техниканың араласуын талап ететін алгоритмдерді есептеу алгоритмдері дейді.Рекурсивті алгоритм деп есептеу алгоритмінің бір түрін айтады. Оның нәтижесі формуланың ішіндегі бір параметрінің мәні басқа бір өзгеріп отыратын параметрден тәуелді болудан шығады.Қосалқы алгоритм дегеніміз күрделі алгоритмдердің бірнеше жай алгоритмгебөлінуі арқылы негізгі алгоритмге қажетті уақытында ғана шақырылатын, жалпылама жағдайға негізделіп дербес құрылатын алгоритмдер.

|  |  |
| --- | --- |
| 7 | Бір өлшемді және екі өлшемді массивтерге қолданушы алгоритмдерін жазып беріңіз |

**Массив** – бір типтегі берілгендер жиынтығы немесе бір атауға біріктірілген айнымалылардың реттелген тізбегі.Айнымалылардың-массив элементтерінің типтпері бірдей болады.Массив бір ғана атпен белгіленеді. Жалпы бірөлшемді  
жиым математикада вектор, ал екіөлшемді  
жиым матрица ұғымдарымен сәйкес келеді. Егер  
жиым атауында бір ғана индекс болса, онда ол  
жиымды бірөлшемді, ал екі индекс болса —  
екіөлшемді және т.с.с. n индекс болса, n өлшемді жиым дейді.

**Мысалы :** [1.6,14.9,-5.0,8.5,0.46] мына сандар **тізбегі бір өлшемді массив** деп,оған А атау беруге болады.Массивтің әр элементі массивтің атымен бегіленеді де,оның индексі қойылады.Массив элементтері индекстері бойынша реттеліп жазылады.Индекс 0 ден басталады.А жиымында 5 элемент бар.

[]-массив осындай тік жақшамен жазылады.

**Бір өлшемді жиымды беру:**

a=[1,2.5,15,-6,3]

print(a)

**Рандом арқылы бір өлшемді массивті экранға шығару**

from random import randint

n=10

a=[0]\*n

for i in range(n):

a[i]=randint(-50, 50)

print(a[i], end=' ')

print(sep='\d')

**Екі өлшемді массивтер** Жиі тапсырмаларда деректермен тікбұрышты кестелерді сақтау керек. Мұндай кестелер матрицалар немесе **екі өлшемді массивтер** деп аталады. Бағдарламалау тілінде Питон кестені жолдар тізімі ретінде ұсынуға болады, оның әрбір элементі өз кезегінде тізім болып табылады, мысалы, сандар. Мысалы, екі жол мен үш бағаннан сандық кестені жасауға болады:

A = [[1, 2, 3], [4, 5, 6]]

Мұнда тізімнің бірінші жолы [0] [1, 2, 3] сандардың тізімі болып табылады. Яғни A[0][0] == 1, мәні A[0][1] == 2, A[0][2] == 3, A[1][0] == 4, A[1][1] == 5, A[1][2] == 6.

**Екі өлшемді массивті енгізу**

Бағдарлама екі өлшемді массивке кіруге мүмкіндік берсін, олардың әрқайсысы бос орындармен бөлінген m сандарды қамтитын N жолдар түрінде. Оларды қалай есептеуге болады? Мысалы, осылай:

A = []

n=3

for i in range(n):

A.append(list(map(int, input().split())))

print(A)

|  |  |
| --- | --- |
| 8 | Шешім қабылдау құрылымдары. Шартты нұсқаулығы if, if-else, if-elif-else. Тасбақа графикасының позициясын, түсін, бұрышын анықтау. |

Шешім қабылдау – ақпарат алмасу сияқты кез келген басќару қызметінің құрамдас бөлігі. Шешім қабылдау қажеттігі басқарушының алға қойған маќсатының қалыптасуы мен сол мақсатқа жетуіндегі іс-қимылын түгелдей қамтиды

Шартты (нұсқаулық) оператор:

if шарт:

1 нұсқаулар блогы

else:

2 нұсқаулар блогы

Шартты оператор келесі алгоритм бойынша жұмыс істейді:

Алдымен IF қызметтік сөзінен кейін орналасқан логикалық өрнектің (шарттың) мәні есептеледі. Егер оның нәтижесі ақиқат болса, жаңа қатарда 4 бос орынмен басталатын 1 нұсқаулар блогы орындалады, егер нәтижесі жалған болса, онда ELSE сөзінен кейін орналасқан 2 нұсқаулар блогы орындалады.

Нұсқаулар блогы қажетті операторлар санынан тұруы мүмкін, бірақ олардың барлығы міндетті түрде 4 бос орыннан кейін жазылуы керек.

1 мысал. Берілген екі санның үлкенін экранға шығару.

x= int(input('Введите первое число'))

y= int(input('Введите второе число'))

if x > y:

print('Большее число', x)

else:

print('Большее число', y)

Шартты нұсқауда else сөзі мен келесі блок болмауы да мүмкін. Мұндай нұсқау толық емес тармақталу деп аталады. Мысалы, егер:

2 мысал. Пернетақтадан х саны еншізіледі және біз оны абсолюттік өлшемге, осы санның модуліне алмастырғымыз келеді. Мұны былай жасауға болады: егер сан теріс болса, «унарлық минус» деген амалды қолдануымыз, ал егер сан оң болса, онда ештеңе өзгертпестен шығаруымыз керек.

x = int(input())

if x < 0:

x = -x

print(x)

Бұл мысалда х айнымалыға –x мәні меншіктеледі, бірақ тек x<0 болғанда ғана. Ал print(x) нұсқауы тексерілетін шартқа тәуелсіз, әрқашанда орындалады.

2 мысал: Пайдаланушы мектептегі сынып нөмірін енгізеді, бағдарлама оның қай сыныпқа жататыны туралы хабарлама шығарады.

n=int(input('Введите номер класса'))

if n<4:

print('Младшие классы')

else:

if n<9:

print('Средние классы')

else:

print('Старшие классы')

Салыстыру операторлары

Әдетте, тексерілетін шарт ретінде төмендегі салыстыру операторларының бірін есептеу нәтижесі қолданылады:

< кіші — шарт дұрыс, егер бірінші операнд екіншісінен кіші болса.

> үлкен — шарт дұрыс, егер бірінші операнд екіншісінен үлкен болса.

<= кіші немесе тең.

>= үлкен немесе тең.

== теңдік. Шарт дұрыс, егер екі операнд тең болса.

!= теңсіздік. Шарт дұрыс, егер екі операнд тең болмаса.

Мысалы, (x\*x < 1000) шарты «x\*x мәні 1000 нан кіші» дегенді, ал (2 \* x != y) шарты «х айнымалысының мәні у айнымалының мәніне тең емес» дегенді білдіреді.

Питонда салыстыру операторларын тізбекке біріктіруге (көптеген басқа да бағдарламалау тілдеріне қарағанда, мұнда ол үшін логикалық байланыстарды қолдану керек) болады, мысалы, a == b == c немесе 1 <= x <= 10.

Сонда 2 мысалды дұрысырақ жазуға болатын еді :

n=int(input('Введите номер класса'))

if 1<=n<=4:

print('Младшие классы')

else:

if 4print('Средняя школа')

else:

print('Старшие классы')

Салыстыру операторлары арнайы bool логикалық типінің мәнін қайтарады. Логикалық типтің мәндері екі мәндердің бірін қабылдауы мүмкін: True (ақиқат) немесе False (жалған). Егер логикалық True мәнін int типіне өзгертсек, онда 1 болады, ал егер False болса, онда 0 береді. Кері жағдайда 0 саны False өзгереді, ал кез келген нөл емес сан True өзгереді. Str типін bool типіне кері өзгерту кезінде, бос қатар False, ал кез келген бос емес қатар True мәніне өзгереді.

Тасбақа графикасының позициясын , түсін , бұрышын анықтау.

Жылы компьютерлік графика, тасбақа графикасы болып табылады векторлық графика туысқанды қолдану меңзер («тасбақа«) а Декарттық жазықтық. Тасбақа графикасы Логотиптің бағдарламалау тілі.

Тасбақаның үш қасиеті бар: орналасу орны, бағдарлау (немесе бағыт) және қалам. Қаламның да атрибуттары бар: түсі, ені және қосу / өшіру күйі.

Тасбақа өз орнына қатысты командалармен қозғалады, мысалы «10 бос орын алға жылжу» және «солға 90 градусқа бұрылу». Тасбақа көтеретін қаламды оны қосу, түсін немесе енін орнату арқылы басқаруға болады. Студент тасбақаның қозғалысын, егер олар тасбақа болса, не істейтіндерін елестету арқылы түсіне алады (және болжап, дәлелдей алады). Сеймур Паперт бұл «дене синтоникалық» ойлау деп атады.

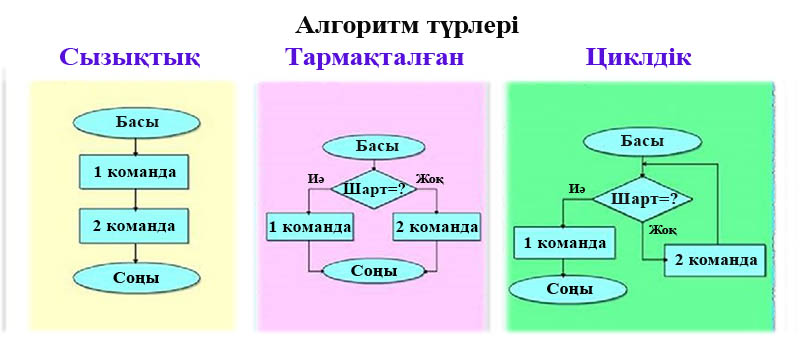
Толық тасбақа графикалық жүйесі басқару ағыны, процедуралар мен рекурсияны қажет етеді: көптеген тасбақаларды сызу бағдарламалары орындалмайды. Осы блоктардан квадраттар, үшбұрыштар, шеңберлер және басқа құрама фигуралар сияқты күрделі пішіндер жасауға болады. Тасбақа графикасының идеясы, мысалы, а Линденмайер жүйесі генерациялау үшін фракталдар.

Тасбақа графикасы көбінесе Логотиптің бағдарламалау тілі.[2] Сеймур Паперт өзінің нұсқасын қолдау үшін 1960 жылдардың соңында Логотипке тасбақа графикасына қолдау көрсетті тасбақа робот, роботтың корпусына орнатылған немесе оған бекітілген кішкене тартылатын қаламды қолданып, оған берілген сызу функцияларын орындауға арналған, пайдаланушының жұмыс бекетінен басқарылатын қарапайым робот. Тасбақаның геометриясы (х,ж) бағытталған Декарттық геометрия, бірінші кезекте векторсияқты координаттық-адрестік жүйелермен салыстырғанда (яғни бастапқы нүктеден салыстырмалы бағыт пен қашықтық) негізделген PostScript. Тәжірибелік мәселе ретінде дәстүрлі модельдің орнына тасбақа геометриясын қолдану тасбақа роботының нақты қозғалыс логикасын имитациялайды. Тасбақа дәстүрлі түрде және көбінесе кескіндемелік түрде үшбұрыш немесе тасбақаның белгішесі түрінде ұсынылады (бірақ оны кез-келген белгішемен көрсетуге болады).

Бүгін Python бағдарламалау тілінің стандартты кітапханасында Turtle графикалық модулі бар.[3] Логотиптің алдыңғы нұсқасы сияқты, Python тасбақасын енгізу бағдарламашыларға екі өлшемді кеңістікте бір немесе бірнеше тасбақаны басқаруға мүмкіндік береді. Стандартты Python синтаксисі, басқару ағыны және деректер құрылымы тасбақа модулімен қатар қолданыла алатындықтан, тасбақа Python оқитын бағдарламашылар үшін тілдің негіздерімен танысудың танымал тәсілі болды.

Тасбақа графикасының идеяларын үш өлшемді кеңістікті қамтуға болады. Бұған бірнеше түрлі координаттар модельдерінің бірін қолдану арқылы қол жеткізіледі. Егер тасбақа цилиндрлік координаттарда жұмыс істейтін болса, онда оның жазықтықта орналасуы мен айдары болады, ал оның жазықтығы тік осьтің айналасында айналуы мүмкін. Бұл көбінесе тасбақаның екі түрлі бағыт бұрышы болатындығынан көрінеді, бірі жазықтықта, екіншісі жазықтықтың бұрышын анықтайды. Әдетте ұшақтың бұрышын өзгерту тасбақа қозғалмайды.

|  |  |
| --- | --- |
| 9 | Циклдік алгоритмдердің сызбаларын құрастырыңыз. Циклдің келесі итерациясына көшу және циклдің орындалуын бұзатын операторларға түсініктеме беріңіз және мысал келтіріңіз. |



Оператор break досрочно прерывает цикл.

Оператор continue начинает следующий проход цикла, минуя оставшееся тело цикла (for или while)

|  |  |
| --- | --- |
| 10 | Іздеу алгоритмдері, максимальды/ минимальды элементті ізде, сызықтық іздеу, бинарлық іздеуді алгоритмін сипаттап жазып беріңіз. |

Іздеу алгоритмі

Барлық әдістерді статикалық және динамикалық деп қарастыруға болады. Массивтен статикалық әдіспен іздеу кезінде оның мәндері өзгермейді. Массивтен динамикалық әдіспен іздеу кезінде оның өлшемі өзгеруі мүмкін, себебі ол қайтадан сұрыпталады. Біз көбінде статикалық әдісті қолданамыз, үйткені мәтіндік редактордағы сөздерді өзгерте алмаймыз, ал динамикалық тәсіл ойын құрғанда пайдаланылады.

Іздеу әдістерін сондай-ақ нақты кілттерді пайдаланатын және туындаушы кілттерді пайдаланатын деп екіге бөледі. Бұл жағдайда кілт деп өзіміз іздеп отырған сөзді айтады. Мәтіндік редакторға қолданылатын кілт – туындаушы болып табылады, себебі ізделінетін массив алдын-ала алфавит бойынша сұрыпталған. Бұл рет іздеуді жеңілдету үшін пайдаланылады.

Кейбір кітаптарда бұл әдіс «экстраполяция әдісі» деп аталады. Экстарполяция – берілген интервалдан тыс бірнәрсені анықтау әдісі, ал интерпояция – сол интервал аралығына анықтау әдісі

максимальды/ минимальды элементті ізде,

from \_\_future\_\_ import division

somelist = [1,12,2,53,23,6,17]

max\_value = max(somelist)

min\_value = min(somelist)

avg\_value = 0 if len(somelist) == 0 else sum(somelist)/len(somelist)

Сызықты іздеудің мағынасы элементтерді тізбекпен таңдап алуды және элементтерді кілт мәнімен салыстырудан тұрады.

Функция парамертлер ретінде массивті, элементтер санын және кілт мәнін алады. Сәйкес элементтің индексін қайталайды, егер іздеу сәтсіз болса, -1 мәнін береді. Тізбектеліп іздеу кез келген тізбек үшін қолайлы, тізбектеліп іздеудің орталық тиімділігі O(n) тең болады.

Бинарлық іздеу.

Бинарлық іздеулер тек қана реттелген тізімдер үшін ғана қолданылады. Мысалы элементтер тұратын массив берілсін. Тізімнің басындағы және соңындағы элементтердің индекстері мынадай low=0 high=n-1 дейін болады. Бинарлық іздеудің алгоритмі:

Массивтің ортаңғы элементінің индексін табу: mid=(low+high)/2.

Орталық элементтің мәнін кілтпен салыстыру «Key». Егер салыстыру нәтижесінде сәйкестік бар болса, онда mid индексін кілтті табу үшін қолданамыз. Егер орталық элемент мәні кілттен кіші болса, онда қарастырылып отырған тізімнің оң жағындағы бөлігінде іздеу жүргіземіз. Егер керісінше үлкен болса, онда сол жақтағы бөлігінде іздеу жүргіземіз.

Егер ізделіп отырған элемент тізімде жоқ болса, онда үзу индикаторын береміз.

|  |  |
| --- | --- |
| 11 | Таңдау арқылы сұрыптаудың және көпіршік сұрыптауының алгоритмін жазып беріңіз және қайсысы тиімдірек . |

Көпіршікті сұрыптау - ең танымал сұрыптау алгоритмдерінің бірі. Мұнда көрші элементтердің мәндерін дәйекті түрде салыстыру керек және алдыңғысы келесіден үлкен болса, сандарды ауыстыру керек. Осылайша, үлкен мәндері бар элементтер тізімнің соңында, ал кіші мәндері бар элементтер басында қалады. Сұрыптаудың бұл түрі информатика пәнімен танысудың басында оқытылады, өйткені ол сұрыптау түсінігін барынша қарапайым түрде көрсетеді. Бұл тәсіл тізімді қайталайды және көрші элементтерді салыстырады. Егер тапсырыс дұрыс болмаса, олар ауыстырылады. Бұл барлық элементтер дұрыс тәртіпте болғанша жалғасады. Көпіршікті сұрыптаудағы қайталаулар санының көп болуына байланысты оның ең нашар күрделілігі O(n^2) болып табылады. Python: def bubble\_sort(arr): def swap(i, j): arr[i], arr[j] = arr[j], arr[i] n = len(arr) swapped = True x = -1 While swapped: swapped = False x = x + 1 for i in range(1, n-x): if arr[i - 1] > arr[i]: swap(i - 1, i) swapped = True Таңдау сұрыптауы іске асыру үшін ең оңай сұрыптау алгоритмі болуы мүмкін. Басқа ұқсас алгоритмдердің көпшілігі сияқты, ол салыстыру операциясына негізделген. Әрбір элементті әрқайсысымен салыстыру және қажет болған жағдайда алмасуды жүзеге асыру әдіс ретті қажетті реттелген пішінге әкеледі. Алгоритм идеясы өте қарапайым. N өлшемді А массиві болсын, онда іріктеу сұрыптауы келесіге дейін азаяды: 1]A[i] тізбегінің бірінші элементін алайық, мұндағы i - элемент нөмірі, бірінші i үшін ол 1-ге тең; 2]тізбектің минималды (максималды) элементін табу және оның нөмірін кілттік айнымалыға сақтау; 3]егер бірінші элементтің нөмірі мен табылған элементтің нөмірі сәйкес келмесе, яғни егер кілт?1 болса, онда бұл екі элемент мәндермен алмасады, әйтпесе ешқандай манипуляциялар болмайды; 4]i-ді 1-ге көбейтіңіз және массивтің қалған бөлігін сұрыптауды жалғастырыңыз, атап айтқанда 2 саны бар элементтен N-ге дейін, өйткені A[1] элементі өз орнын алып қойған; Әрбір келесі қадаммен алгоритм жұмыс істейтін ішкі жиымның өлшемі 1-ге азаяды, бірақ бұл сұрыптау әдісіне әсер етпейді, ол әрбір қадам үшін бірдей. Eң нашар уақыт O(n2) Ең жақсы уақыт O(n2) Орташа уақыт O(n2) Жад құны O(n) барлығы, O(1) қосымша Python: def swap(arr, i, j): arr[i], arr[j] = arr[j], arr[i] def select\_sort(arr): i = len(arr) while i > 1: max = 0 for j in xrange(i): if arr[j] > arr[max]: max = j swap(arr, i - 1, max) i -= 1 Қарапайым Θ(n 2) орташа регистрлік алгоритмдердің ішінде таңдау сұрыптауы әрқашан дерлік көпіршікті сұрыптау мен гномды сұрыптаудан асып түседі, бірақ әдетте кірістіру сұрыптауынан асып түседі. Кірістіру сұрыптауы өте ұқсас, себебі k-итерациядан кейін массивтегі бірінші k элементтер сұрыпталған тәртіпте болады. Кірістіру сұрыптасының артықшылығы мынада, ол k + 1-ші элементті орналастыру үшін қажет болғанша көп элементтерді сканерлейді, ал таңдау сұрыптауы k + 1-ші элементті табу үшін барлық басқа элементтерді сканерлеуі керек.

|  |  |
| --- | --- |
| 12 | Сұрыптау алгоритмдерінің мақсатын түсіндіріп беріңіз. Таңдау бойынша сұрыптау алгоритмін жазып беріңіз. |

Сұрыптау дегеніміз—берілген жиынның элементтерін белгілі бір ережелерге сәйкес орналастыру.

Оның негізгі көздеген мақсаты – сұрыпталған жиыннан керек элементтерді іздеуді жеңілдету. Сұрыптауды көбіне массивтерді және файлдарды сұрыптағанда көп қолданады. Бұл екеуін әдетте ішкі және сыртқы сұрыптаулар деп атайды. Массивтер “ішкі” (жедел) жадыда орналасатындықтан, ішкі сұрыптау болады. Бұл жадыға тез қатынаймыз, ал файлдар бұдан бәсеңдеу, бірақ сыйымдылығы үлкендеу “сыртқы” жадыда, яғни есте сақтау құрылғыларында (диск, лента т.б.) сақталатындықтан, оны сыртқы сұрыптау деп атаймыз.

Сұрыптаудың Шелл әдісі бойынша сұрыптау, Хоар әдісі бойынша сұрыптау, таңдап сұрыптау сияқты түрлері бар.

Соның ішіндегі **Таңдау арқылы сұрыптау** - бұл сұрыптаудың ең қолайлы түрі. Әдетте бұл әдіс кестені реттеуді қажет еткен адам ойына ең бірінші келеді. Бұның мәні мынада, мысалы **n** элементтен тұратын **А** сандар массиві берілген. Оны таңдау әдісін қолданып элементтерінің өсуі бойынша сұрыптау қажет.

|  |  |
| --- | --- |
| 13 | Санау арқылы сұрыптау және Жылдам сұрыптау алгоритмдерін сипаттап беріңіз. |

13-сұрақ Санау арқылы сұрыптау алгоритмі - тізімнің элементтерін белгілі бір тәртіпке, мысалы, ең төменнен жоғарыға, ең төменнен төменге, өсіп келе жатқан тәртіпке, азайту тәртібіне, алфавиттік т.с.с. ұйымдастыруға арналған әдіс. сандық және лексикографиялық тәртіпті құрайды. Алгоритмдер негізгі сұрыптау ретінде сұрыптауды жиі қолданады. Сандық сұрыптау алгоритмдерінің алуан түрлері бар, олардың әрқайсысында көптеген әдістер қолданылады. Осындай танымал, бірақ қуатты алгоритмнің бірі - көп тармақты рекурсияға негізделген алгоритм болып табылатын Divide және Conquer алгоритмі. Жылдам сұрыптау және біріктіру сұрыптау - бөлу және жеңу алгоритміне негізделген екі жиі қолданылатын алгоритмдер. Жылдам сұрыптау - бөлу және жеңу тәсіліне негізделген жоғары тиімді, бірақ тиімді сұрыптау алгоритмі, ол екі немесе одан да көп ішкі мәселелерге бөлініп, содан кейін рекурсивті түрде шешілетін динамикалық тәсілге өте ұқсас. Егер қосалқы есептердің мөлшері аз болса, онда олар еш қиындықсыз қарапайым түрде шешіледі. Жылдам сұрыптау алгоритмі бөлімді айырбастау сұрауы деп аталады, тізімді үш негізгі бөлікке бөледі: 1) жиынтық элемент (орталық элементтер), 2) айналмалы элементтерден аз элементтер және 3) айналмалыдан үлкен элементтер. Біліктің өзі екі топтың арасына өзінің соңғы орнына ауысады, содан кейін САҚТАНДЫҚ СОРТ рекурсивті қолданылады.

|  |  |
| --- | --- |
| 14 | Пирамидалық сұрыптау және біріктіру сұрыптауының алгоритмін сипаттап беріңіз. |

|  |  |
| --- | --- |
|  |  |

**Сұрыптау** *(Сортировка; sorting)* - массив элементтерін белгілі бір заңдылықпен орындарын ауыстырып реттеупроцессін айтамыз. Мысалы, сандар массивін өсуі, кемуі бойынша сұрыптау, жолдар массивін алфавит бойынша сұрыптау және тағы басқа. Сұрыптау мақсаты - көптеген сұрыпталған обьектінің ішінен белгілі бір элементті іздеуді оңайлату. Ақпараттық жүйелерде мәліметтерді сұрыптаудың маңызы өте зор.

**Пирамидалық сұрыптау алгоритмі** таңдаумен сұрыптау алгоритмі классына жатады. Бірақ бұл алгоритмнің өзінің ерекше қасиеті бар. Біріншіден, бастапқы массив бинарлы ағашпен беріледі. Бұл бинарлы ағаш құрылысы жағынан ерекшеленіп келеді, ондай ағаштарды толықтау ағаштар деп атайды.

Реттеу процессі екі этаптан тұрады. Бірінші этапта пирпмида тәріздес ағаш түзіледі. 1-этаптың нәтижесі келесі қасиетге ие пирамида болады: ағаш түбірінен қорытынды төбеге шығатын кез-келген жолында түйіндер кемімелі, дәлелдеп айтқанда өспелі емес ретпен орналасады; пирамида ағаштың жоғарыдан төменге, сол жақтан оңға қарай түзіледі. 2-этапта құрылған ағаш-пирамидасында түйінд/іне бекітілген.

Массив элементтеріне түпкілікті реттеуге пайдаланылады. Бұл этапта пирамиданың элементтерін реттеу бағыты пирамиданың оң жағынан солға қарай, төменнен жоғарыға қарай қажет болған жағдайда алмастырылады. Екінші этап бірінші фазадан тұрады. Бірінші фазада массивтің реттелмеген бөлігіндегі ағымдық максималды элемент (бұл фазада оның пирамиданың төбесінде орналасқанын ұмытпау керек) пирамиданың төменгі оң бұрышында орналасқан элементпен алмастырылады. Сөйтіп ағымдық максималды элемент реттелген массивте өзінің орнын табады. Осыдан кейін бұл максималды элемент те, оның орны да реттеуге қарастырылмайды. 2-этаптың 1-фазасының нәтижесінде пирамиданың төбесінде, яғни ағаштың түбірінде, оның құрылысының бұзылуына кездесеміз. Сондықтан 2-фазада өзгенген пирамиданың құрылысы қайтадан құралады.

**Алгоритмдерді біріктіру** отбасы болып табылады алгоритмдер бірнеше алады сұрыпталған кірістер ретінде тізімдейді және барлық тізімдер элементтерін сұрыпталған тәртіппен қамтитын шығыс ретінде бір тізімді шығарады. Бұл алгоритмдер ретінде қолданылады ішкі бағдарламалар әртүрлі сұрыптау алгоритмдері, ең әйгілі біріктіру сұрыптау.

Біріктіру алгоритмі шешуші рөл атқарады біріктіру алгоритм, а салыстыруға негізделген сұрыптау алгоритмі. Тұтастай алғанда сұрыптау алгоритмі екі кезеңнен тұрады:

Рекурсивті тізімді (шамамен) бірдей ұзындықтағы қосалқы тізімдерге бөлу, әр қосалқы тізімде тек бір ғана элемент болғанша немесе итеративті (төменнен жоғары) біріктіру сұрыпталған жағдайда, тізімін қарастырыңыз n сияқты элементтер n өлшемді ішкі тізімдер 1. Бір элементтен тұратын тізім, анықтамалық бойынша, сұрыпталған.

Бірыңғай тізім барлық элементтерді қамтығанша жаңа сұрыпталған қосалқы тізімді жасау үшін қосалқы тізімдерді бірнеше рет біріктіріңіз. Жалғыз тізім - бұл сұрыпталған тізім.

Біріктіру алгоритмі біріктіруді сұрыптау алгоритмінде бірнеше рет қолданылады.

Біріктіру сұрыптауының мысалы мысалда келтірілген. Ол 7 бүтін саннан тұратын сұрыпталмаған массивтен басталады. Массив 7 бөлімге бөлінген; әр бөлімде 1 элемент бар және сұрыпталған. Содан кейін сұрыпталған бөлімдер үлкенірек, сұрыпталған бөлімдерді шығару үшін біріктіріледі, 1 бөлім, сұрыпталған массив қалғанға дейін.

|  |  |
| --- | --- |
| 15 | Рекурсивті алгоритмдер, рекурсивті қатынас, рекурсия, рекурсивті бөліну, рекурсия қауіптілігін түсіндіріп беріңіз. |

Рекурсия. Рекурсивті алгоритмдер

Өзін өзі шақыратын функция рекурсия деп аталады. Рекурсия тереңдігі дегеніміз – функция мәнін есептеуде өзін-өзі шақыру саны. Рекурсивті программалау стек принципіне сүйенеді.

Рекурсия түрлері мынадай:

сызықтық рекурсия;

параллель рекурсия;

қосалқы рекурсия;

жоғары ретті рекурсия.

Рекурсивті функциялар

Ары қарай қарастыру үшін бізге бірқатар анықтамалар қажет болады. Айталық, Х және Ү екі жиыны бар болсын.

Егер Х жиынының кейбір элементтеріне Ү жиының бірмәнді анықталған элементтері сәйкестендірілген болса, онда Х-тан Ү-ке жекелеме функция (частичная функия) берілген деп аталады. Ү-те сәйкес элементтері бар Х-тің элементтерінің жиынтығы функияның анықталу облысы деп аталады, ал Х-тің элементтеріне сәйкес келетін Ү-тің элементтерінің жиынтығы функия мәндерінің жиынтығы деп аталады. Егер функцияның Х-тен Ү-ке анықталу облысы Х жиынымен сәйкес келсе, онда функция барлық жерде анықталған деп аталады.

Рекурсивті функия ұғымына сүйеніп алгоритм ұғымын нақты анықтамасын құрудың бастапқы идеясы мынада жатыр: кез-келген берілгендерді (әрине дискретті) қандай-да бір санау жүйесінде натурал сандармен кодтауға болады, және сонда оларды кез-келген жолмен түрлендіру - есептеу операцияларының тізбегіне келтіріледі, ал өңдеу нәтижесі де сол сияқты бүтін санды

Бұл жағдайда осы сандық функцияға бірдей кез-келген алгоритм оның мәнін есептейді, ал оның элементар қадамдары кәдімгі арифметикалық және логикалық амалдар болып табылады. Мұндай функциялар есептелінетін функциялар деп аталады.

Рекурсивті алгоритмдер мысалдары:

6!- алты факториялды есептеу алгоритмі:

#include

double fact(int n);

int n=6;

double f;

f=fact(n);

printf(‘6!=%10.0f\n”,f);

return (0);

}

double fact(int n)

{

if (n<1) return(1.0)

else

return(n\*fact(n-1);

}

Евклид алгоритмі бойынша екі санның ең үлкен ортақ бөлгіші табу

алгоритмі:

#include

#include

double euob(double n, double m);

int main ()

{

double f;

double n=1470;

double m=693;

f=euob(n,m);

printf("f=%10.0f \n”, f);return(0);

}

double euob(double n, double m)

{

w=floor(fmod(n,m);

if (w==0) return(m);

else return(euob(m,w));

}

|  |  |
| --- | --- |
| 16 | Динамикалық деректер құрылымы, стек, кезек, декті сипаттап беріңіз. |

16. Динамикалық деректер құрылымы, стек, кезек, декті сипаттап беріңіз

Динамикалық деректер құрылымдары-бұл жад қажет болған жағдайда бөлінетін және босатылатын мәліметтер құрылымы.   
Жадта өмір сүру процесінде деректердің динамикалық құрылымдары олардың құраушы элементтерінің санын ғана емес, сонымен қатар элементтер арасындағы қатынастардың сипатын да өзгерте алады. Бұл деректер элементтерінің мазмұнын өзгертуді ескермейді. Динамикалық құрылымдардың олардың мөлшерінің және элементтер арасындағы қатынастардың табиғатының сәйкес келмеуі сияқты ерекшелігі, машиналық кодты құру кезеңінде компилятор бағдарламасы бүкіл құрылым үшін белгіленген көлемдегі жад бөлігін бөле алмайды, сонымен қатар нақты мекен-жайларды құрылымның жеке компоненттерімен салыстыра алмайды. .

Динамикалық деректер құрылымы мыналармен сипатталады:

* оның аты жоқ;
* бағдарламаны орындау барысында оған жад бөлінеді;
* Құрылым элементтерінің саны бекітілмеуі мүмкін;
* Бағдарламаның орындалу барысында құрылымның өлшемі өзгеруі мүмкін;
* Бағдарламаны орындау барысында құрылым элементтері арасындағы өзара байланыстың сипаты өзгеруі мүмкін.

Деректердің әрбір динамикалық құрылымы көрсеткіш түрінің статикалық айнымалысын (оның мәні-осы объектінің мекен-жайы) салыстырады, ол арқылы динамикалық құрылымға қол жеткізуге болады.   
Динамикалық құрылымдар, анықтама бойынша, жадтағы Құрылым элементтерінің физикалық іргелес болмауымен, оны өңдеу процесінде құрылым мөлшерінің (элементтер санының) тұрақсыздығымен және болжанбауымен сипатталады.  
Динамикалық деректер құрылымын жіктеу   
көптеген тапсырмалар бағдарламаны орындау барысында конфигурациясы, өлшемдері мен құрамы өзгеруі мүмкін деректерді пайдалануды талап етеді. Оларды ұсыну үшін динамикалық ақпараттық құрылымдар қолданылады. Мұндай құрылымдарға мыналар жатады:

* бір бағытты тізім;
* көп бағытты тізім;
* циклдік тізімдер;
* стек;
* дек;
* кезек;
* екілік ағаштар.

Олар жеке элементтерді және рұқсат етілген операцияларды байланыстыру әдісімен ерекшеленеді. Динамикалық құрылым жедел жадтың іргелес емес бөліктерін иелене алады.  
  
 **Стек** (ағылш. stack-стек) - Бұл элементтердің реттелген жиынтығы болып табылатын мәліметтер құрылымы, онда жаңа элементтерді қосу және барын жою стек шыңы деп аталатын бір ұшынан жасалады. Сонымен қатар, стектің біріншісі оған соңғы орналастырылған элементті алып тастайды, яғни стекте "соңғы кірді — бірінші шықты" стратегиясы жүзеге асырылады (соңғы-in, бірінші-out — LIFO). Шынайы өмірдегі стектің мысалы табақтар жиынтығы болуы мүмкін: табақты шығарғымыз келгенде, жоғарыдағы барлық тақтайшаларды алып тастауымыз керек. Стек операцияларының сипаттамасына оралыңыз:

1. emptyempty-стек элементтерін тексеру,
2. pushpush (стекке жазу) — жаңа элементті енгізу әрекеті,
3. poppop (дестеден шығару) — жаңа элементті жою әрекеті.

**Дек** (ағылш. deque-double ended queue) — жаңа элементтерді қосу және бар элементтерді жою екі ұшынан жасалатын элементтер тізімін білдіретін деректер құрылымы. Бұл құрылым FIFO-ны да, LIFO-ны да қолдайды, сондықтан оны стек те, кезек те жүзеге асыра алады. Бірінші жағдайда сіз тек бастың немесе құйрықтың әдістерін, екіншісінде — екі түрлі ұштың итеру және pop әдістерін қолдануыңыз керек. Деканы екі жақты кезек ретінде қабылдауға болады. Оның келесі операциялары бар:

1. emptyempty-элементтерді тексеру,
2. pushBackpushBack — соңына жазу) - жаңа элементті соңына енгізу операциясы,
3. popBackpopBack (соңынан шығару) — соңғы элементті жою әрекеті,
4. pushFrontpushFront — басына жазу) - жаңа элементті басына енгізу операциясы,
5. popFrontpopFront (басынан бастап алып тастау) — бастапқы элементті жою әрекеті.

**Кезек** (ағыл. queue) - бұл мәліметтер құрылымы, элементтерді қосу және жою сәйкесінше pushpush және poppop операциялары арқылы жүзеге асырылады. Сонымен қатар, бірінші кезекте алдымен орналастырылған элемент жойылады, яғни кезекте "бірінші кірді — бірінші шықты" принципі жүзеге асырылады (ағылш. first-in, first-out — FIFO). Кезектің басы бар (ағылш. head) және құйрық (ағылш. tail). Элемент кезекке қойылған кезде, ол құйрығында орын алады. Оның басында тұрған элемент әрқашан кезектен шығады. Кезек келесі әрекеттерді қолдайды:

1. emptyempty-ондағы элементтердің кезегін тексеру,
2. pushpush (кезекке жазу) - жаңа элементті енгізу әрекеті,
3. poppop (кезектен шығару) - жаңа элементті жою әрекеті,
4. sizesize-кезекте элементтер санын алу операциясы.

|  |  |
| --- | --- |
| 17 | Байланыстырылған тізім және қосарланған тізімге түсініктеме беріңіз. |

|  |  |
| --- | --- |
| 18 | Тізімдерді массивтер арқылы іске асырыңыз. Нұсқауыштар арқылы тізімдерді жүзеге асыр,мысал келтіре отырып. |

Бұрын біз тізбекті элементтерді өңдеу міндетіне тап болдық, мысалы, тізбектің ең үлкен элементін есептеу. Бірақ біз компьютердің жадында барлық тізбекті сақтамадық, алайда, көптеген тапсырмаларда, мысалы, біз өсіп келе жатқан тәртіпте тізбектің барлық элементтерін шығару қажет болса, бүкіл тізбекті сақтау қажет ("тізбекті сұрыптау").

Мұндай деректерді сақтау үшін Питонда тізім деп аталатын деректер құрылымын пайдалануға болады (көптеген бағдарламалау тілдері басқа "массив" термині қолданылады). Тізім-жолдағы таңбалар ретінде 0-ден нөмірленген элементтердің тізбегі. Тізімді квадрат жақшада тізім элементтерін тізіммен көрсетуге болады, мысалы, тізімді осылай көрсетуге болады:

Primes = [2, 3, 5, 7, 11, 13]

Rainbow = ['Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Indigo', 'Violet']

|  |  |
| --- | --- |
| 19 | Графтағы алгоритмдер, граф жайлы түсініктеме беріңіз. Терең іздеу, ені бойынша іздеу, дейкстра алгоритмін сипаттап беріңіз. |

|  |  |
| --- | --- |
| 20 | Ағаштар жайлы негізгі түсінік беріңіз. Бинарлы ағаш, екілік үйінді, ағаш теңдестіру, қызыл-қара ағашқа түсініктеме беріңіз. |

Екілік ағаш-бұл иерархиялық мәліметтер құрылымы, онда әр түйіннің екіден көп емес ұрпағы болады. Әдетте, біріншісі ата-аналық түйін деп аталады, ал балалар сол және оң мұрагерлер деп аталады. Екілік ағаш-реттелген бағдарланған ағаш.

Екілік үйінді-бұл келесі қасиеттері бар екілік ағаш.

1) бұл толық ағаш (барлық деңгейлер толығымен толтырылған, мүмкін соңғы деңгейден басқа, ал соңғы деңгейде барлық кілттер сол жақта қалады). Екілік үйінділердің Бұл қасиеті оларды массивте сақтауға жарамды етеді.

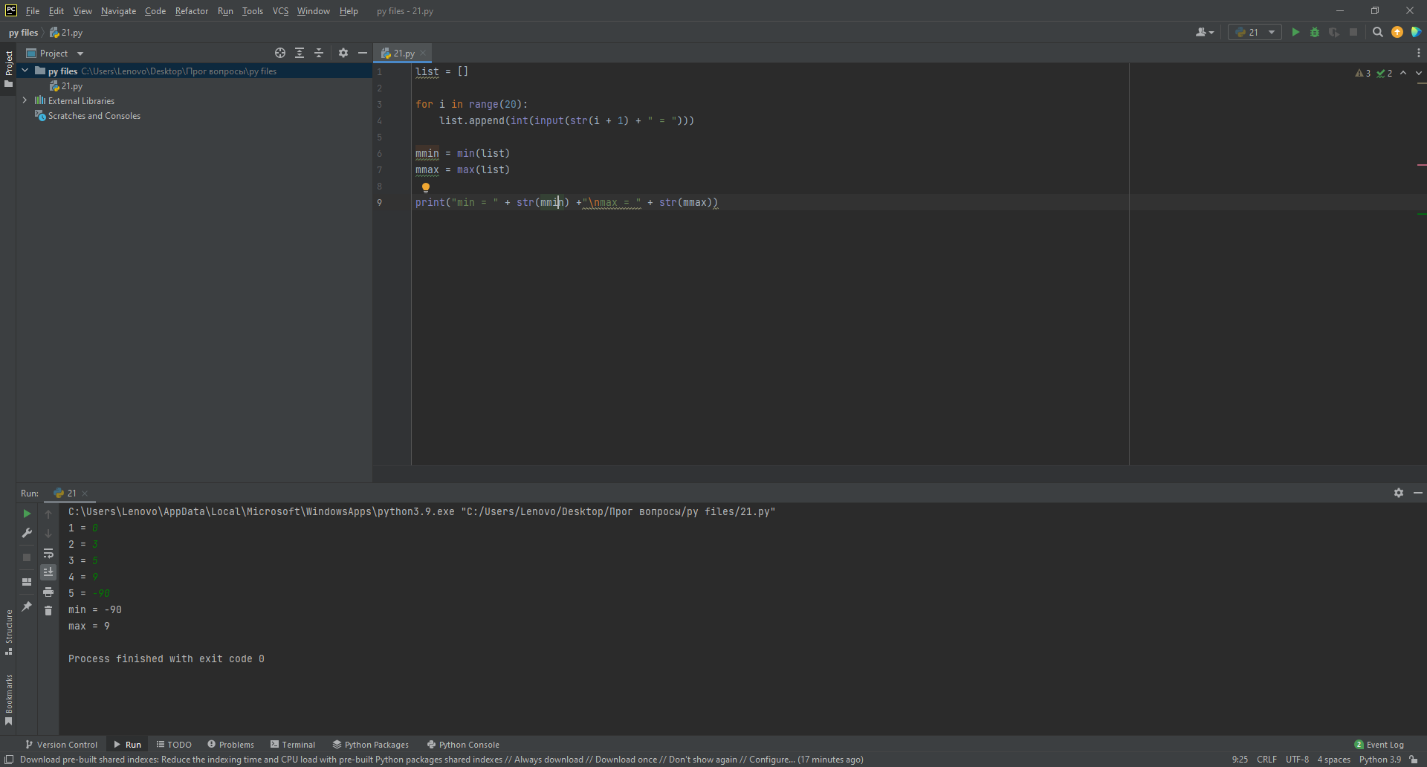
2) екілік қадалар-бұл минималды қадалар немесе максималды қадалар. Минималды екілік қадада түбірлік кілт екілік үйіндідегі барлық кілттердің арасында минималды болуы керек. Екілік ағаштың барлық түйіндері үшін бірдей қасиет рекурсивті түрде шынайы болуы керек.

Теңдестірілген екілік ағаш мәселесінде а-ның түбірін келтірдік екілік ағаш. Бұл биіктік тепе-теңдігі екенін анықтауға тура келеді

Қызыл қара ағаш бинарлы іздеу ағашы, онда теңестіру ағаш түйінінің «түсі» негізінде жүзеге асады, ол тек қана екі мән қабылдайды «қызыл» және «қара». Сонымен қатар ағаштың барлық жапырақтары бос болады және оларда мәлімет болмайды, бірақ ол ағашқа қатысты болады әрі қара деп есептеледі.

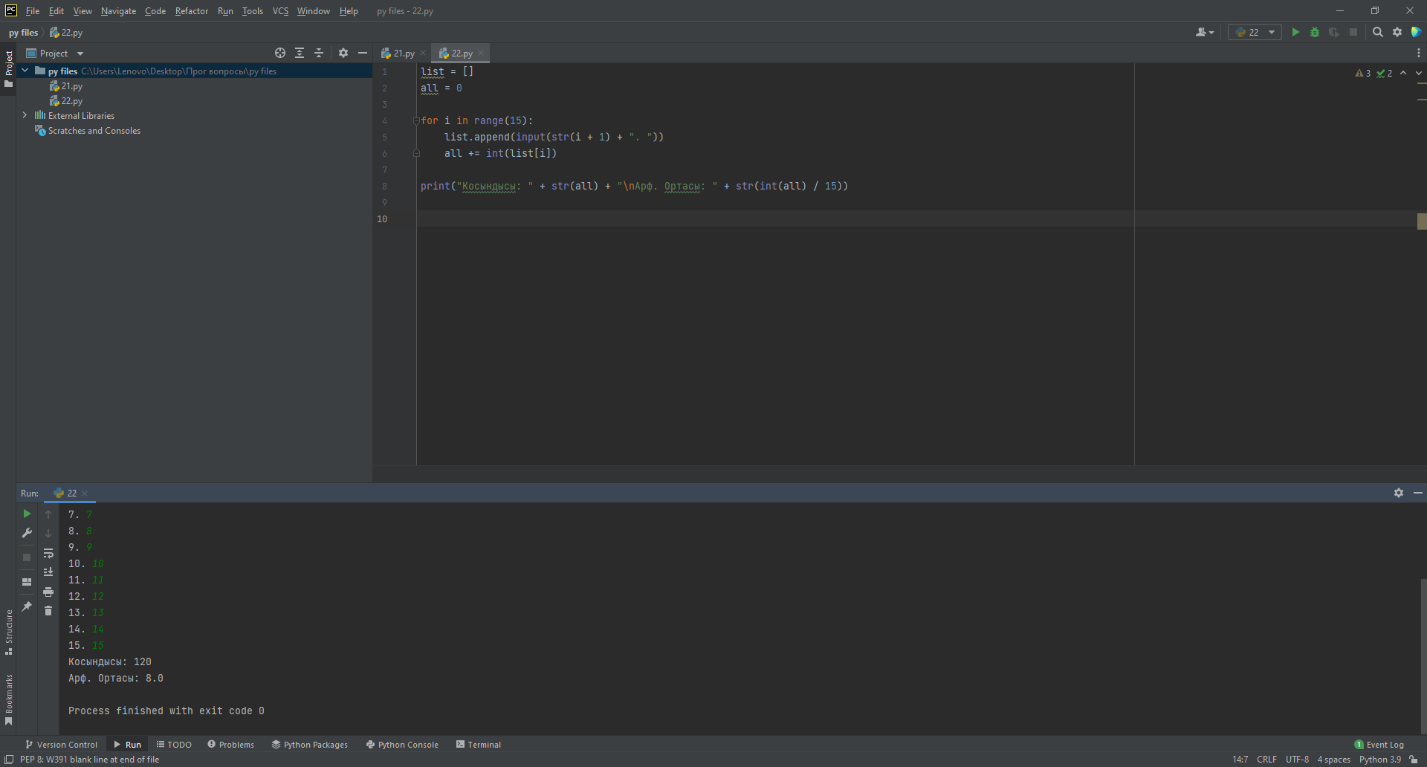
|  |  |
| --- | --- |
| 21 | Пайдаланушыдан 20 саннан тұратын қатарды енгізуді сұрайтын алгоритм құрастырыңыз. Алгоритмде сандарды тізімде сақтауы керек, содан кейін келесі деректерді көрсетуі керек:  • тізімдегі ең кіші сан;  • тізімдегі ең үлкен сан; |

list = []  
  
for i in range(20):  
 list.append(int(input(str(i + 1) + " = ")))  
  
mmin = min(list)  
mmax = max(list)  
  
print("min = " + str(mmin) +"\nmax = " + str(mmax))



|  |  |
| --- | --- |
| 22 | Пайдаланушыдан 15 саннан тұратын қатарды енгізуді сұрайтын алгоритмді құрастырыңыз. Алгоритм сандарды тізімде сақтауы керек, содан кейін келесі деректерді көрсетуі керек:  • тізімдегі сандардың қосындысы;  • тізімдегі сандардың арифметикалық ортасы. |

list = []  
all = 0  
  
for i in range(15):  
 list.append(input(str(i + 1) + ". "))  
 all += int(list[i])  
  
print("Косындысы: " + str(all) + "\nАрф. Ортасы: " + str(int(all) / 15))



|  |  |
| --- | --- |
| 23 | n нақты элементтерден тұратын бір өлшемді жиымда: жиымның ең үлкен элементінен кейін орналасқан элементтерінің қосындысын анықтайтын алгоритм құрыңыз |

from random import randint  
  
n = int(input("Enter N:"))  
a = []  
for i in range(n):  
 a.append(randint(1,100))  
print("Original array:", a)  
m = max(a)  
s = 0  
r = []  
for i in range(n):  
 if a[i] == m:  
 r = a[i+1:n]  
print("Elements after the maximum element:",r)  
for i in r:  
 s += i  
print("Sum:", s)

|  |  |
| --- | --- |
| 24 | 25 нақты кездейсоқ элементтерден тұратын бір өлшемді жиымды көпіршік әдісін пайдаланып сұрыптаңыз. |

from random import randint

def bubble(array):

for i in range(N-1):

for j in range(N-i-1):

if array[j] > array[j+1]:

buff = array[j]

array[j] = array[j+1]

array[j+1] = buff

N = 25

a = []

for i in range(N):

a.append(randint(1, 99))

print(a)

bubble(a)

print(a)

|  |  |
| --- | --- |
| 25 | 18 нақты элементтерден тұратын бір өлшемді жиымды таңдау әдісі арқылы пайдаланып сұрыптаңыз. |

25.18 нақты элементтерден тұратын бір өлшемді жиымды таңдау әдісі арқылы пайдаланып сұрыптаңыз.

# Заполняем список из 18 элементов

# случайными числами от 1 до 99 и

# выводим неотсортированный список на экран.

from random import randint

N = 18

arr = []

for i in range(N):

    arr.append(randint(1, 99))

print(arr)

# В цикле переменная i хранит индекс ячейки,

# в которую записывается минимальный элемент.

# Сначала это будет первая ячейка.

i = 0

# N - 1, так как последний элемент

# обменивать уже не надо.

while i < N - 1:

    # ПОИСК МИНИМУМА

    # Сначала надо найти минимальное значение

    # на срезе от i до конца списка.

    # Переменная m будет хранить индекс ячейки

    # с минимальным значением.

    # Сначала предполагаем, что

    # в ячейке i содержится минимальный элемент.

    m = i

    # Поиск начинаем с ячейки следующей за i.

    j = i + 1

    # Пока не дойдем конца списка,

    while j < N:

        # будем сравнивать значение ячейки j,

        # со значением ячейки m.

        if arr[j] < arr[m]:

            # Если в j значение меньше, чем в m,

            # сохраним в m номер найденного

            # на данный момент минимума.

            m = j

        # Перейдем к следующей ячейке.

        j += 1

    # ОБМЕН ЗНАЧЕНИЙ

    # В ячейку i записывается найденный минимум,

    # а значение из ячейки i переносится

    # на старое место минимума.

    arr[i], arr[m] = arr[m], arr[i]

    # ПЕРЕХОД К СЛЕДУЮЩЕЙ НЕОБРАБОТАННОЙ ЯЧЕЙКЕ

    i += 1

# Вывод отсортированного списка

print(arr)

|  |  |
| --- | --- |
| 26 | 20 нақты элементтерден тұратын бір өлшемді жиымды жылдам сұрыптау әдісін пайдаланып сұрыптаңыз. |

import random

from random import randint

n=20

a=[]

for i in range (n):

a.append(randint(1,99))

print(a)

def quick\_sort(s):

if len(s)<=1:

return s

elem=s[0]

left=list(filter(lambda x: xelem,s))

return quick\_sort(left)+center+quick\_sort(right)

print (quick\_sort(a))

|  |  |
| --- | --- |
| 27 | 25 нақты элементтерден тұратын бір өлшемді жиымды пирамидалық сұрыптау әдісін пайдаланып сұрыптаңыз. |

def heapify(arr, n, i):

largest = i

l = 2 \* i + 1

r = 2 \* i + 2

if l < n and arr[i] < arr[l]:

largest = l

if r < n and arr[largest] < arr[r]:

largest = r

if largest != i:

arr[i],arr[largest] = arr[largest],arr[i]

heapify(arr, n, largest)

def heapSort(arr):

n = len(arr)

for i in range(n // 2 - 1, -1, -1):

heapify(arr, n, i)

for i in range(n-1, 0, -1):

arr[i], arr[0] = arr[0], arr[i]

heapify(arr, i, 0)

arr = [25,13,23,2,9,20,19,18,6,16,15,5,24,12,11,10,21,8,7,17,14,4,3,22,1,]

heapSort(arr)

n = len(arr)

print ("Sorted array is")

for i in range(n):

print ("%d" %arr[i]),

|  |  |
| --- | --- |
| 28 | 15 нақты элементтерден тұратын бір өлшемді жиымды кірістіру әдісін пайдаланып сұрыптаңыз. |

def insert\_sort(nums):

for i in range(1, len(nums)):

item\_to\_insert = nums[i]

j = i - 1

while j >= 0 and nums[j] > item\_to\_insert:

nums[j + 1] = nums[j]

j -= 1

nums[j + 1] = item\_to\_insert

arr = [9, 1, 15, 28, 6]

insert\_sort(arr)

print(arr)

|  |  |
| --- | --- |
| 29 | n нақты элементтерден тұратын бір өлшемді жиымда: жиымның 12 элементін нөмірін бинарлы іздеу әдісі арқылы табыңыз. |

import random

n = 15

mass = [round(random.random(), 2) for k in range(n)]

print(mass)

def bisearch(arr, key):

for i in range(len(arr)):

if arr[i] == key:

return i

return -1

for j in range(12):

print(f"{bisearch(mass, mass[j])}-ый: {mass[j]} ", end="\t")

|  |  |
| --- | --- |
| 30 | Жауын-шашын статистикасы. Пайдаланушыға 12 айдың әрқайсысы үшін жауын-шашынның жалпы мөлшерін тізімге енгізу. Жылына жауын-шашынның жалпы мөлшерін, ай сайынғы жауын-шашынның орташа мөлшерін және  ең жоғары айларды есептеп,  жауын-шашынның ең аз мөлшерін анықтайтын алгоритмін құрыңыз. |

list = []  
n = 12 #ай  
all = 0 #общ  
arf = 0 #арифметикалык ортасы  
  
for i in range(n):  
 list.append(int(input(str(i+1) + ". ")))  
 all += list[i]  
  
print("Жалпы молшеры: " + str(all))  
  
for i in range(n):  
 buffer\_list = 0  
 buffer = 0  
 while(buffer <= i):  
 buffer\_list += list[buffer]  
 buffer += 1  
 print(str(i + 1) + "-шы айдагы жауын-шашыннын орташа молшеры" + str(buffer\_list / (i + 1)))  
  
print("Жауын шашынын ен аз молшеры: " + str(min(list)))

|  |  |
| --- | --- |
| 31 | Алгоритмдердің түсінігі мен қасиеттеріне тоқталыңыз. Базалық алгоритмдік конструкциялар туралы жазыңыз. |

Алгоритмдік конструкциялар

Информатика курсынан бұрыннан белгілі болғандай, кез келген алгоритмді үш негізгі алгоритмдік құрылым арқылы жазуға болады: орындалу, цикл және тармақ.

Келесі алгоритмдік құрылысты құрайтын командалар жазылу реті бойынша бірінен соң бірі тізбектеліп орындалады. Цикл және тармақ командалары программадағы басқа командалардың орындалу ретін басқарады және басқару командалары (басқару конструкциялары) деп аталады.

Алгоритмдік тармақталу конструкциясы белгілі бір шарттың ақиқаттығына немесе жалғандығына байланысты сол немесе басқа командалар тізбегінің орындалуын қамтамасыз етеді. Тармақ операторы – программалау тілінде алгоритмдік тармақталу құрылымын жүзеге асыратын команда.

Алгоритмдік құрастырудың қайталануы (цикл) – қайталап орындалатын әрекеттер тізбегі. Тізбектің өзі цикл денесі деп аталады. Цикл операторы – бағдарламалау тілінде алгоритмдік қайталау құрылымын жүзеге асыратын команда.

Циклдің денесі қанша рет қайталанатынын бақылаудың әртүрлі жолдары бар. Циклды жалғастыру немесе тоқтату шартын, сонымен қатар цикл денесінің қайталану санын орнатуға болады. Осыған байланысты алғы шарты бар цикл, кейінгі шарты бар цикл және параметрі бар цикл бөлінеді.

Циклдің денесі қанша рет қайталанатынын бақылаудың әртүрлі жолдары бар. Циклды жалғастыру немесе тоқтату шартын, сонымен қатар цикл денесінің қайталану санын орнатуға болады. Келесі циклдар ажыратылады: алғы шарты бар цикл, кейінгі шарты бар цикл және параметрі бар цикл. Циклды таңдау тапсырмаға байланысты. Көптеген жағдайларда циклдар өзара ауыстырылады. Циклды таңдағанда, сіз мыналарға назар аудара аласыз:

* қайталану саны белгілі болған кезде параметрі бар цикл қолданылады;
* алғы шарты бар цикл жұмысты жалғастыру шарты белгілі болған кезде қолданылады;
* постшарт циклі цикл денесі кемінде бір рет орындалуы керек кезде қолданылады.

|  |  |
| --- | --- |
| 32 | Алгоритмдердің күрделілігі. IDLE қосымша мүмкіндіктерін ашып көрсетіңіз. |

IDLE — Python интеграцияланған әзірлеу және оқу ортасы.

IDLE келесі мүмкіндіктерге ие:

-tkinter GUI құралдар жинағы арқылы 100% таза Python тілінде кодталған

-Кросс-платформа: Windows, Unix және macOS жүйелерінде бірдей дерлік жұмыс істейді

Python қабықшасының терезесі (интерактивті интерпретатор) кіріс бояуы, код шығысы және қате туралы хабарлар

-Бірнеше қайтару, Python бояуы, смарт шегініс, қоңырау туралы кеңестер, автоматты аяқтау және т.б. бар көп терезелі мәтіндік редактор

кез келген терезеде іздеу, редактор терезелерінде ауыстыру және бірнеше файлдарда іздеу (grep)

-Тұрақты тоқтау нүктелері бар отладчик, жаһандық және жергілікті аттар кеңістігінен өту және қарау

-Конфигурация, браузерлер және басқа диалогтар

IDLE терезелердің екі негізгі түрі бар: Shell терезесі және редактор терезесі. Бір уақытта бірнеше редактор терезелері болуы мүмкін. Windows және Linux жүйелерінде әрқайсысының жеке жоғарғы мәзірі бар. Төменде құжатталған әрбір мәзір оның қай терезе түрімен байланысты екенін анықтайды.

Өңдеу => Файлдардан табу сияқты шығыс терезелері редактор терезесінің ішкі түрі болып табылады. Қазіргі уақытта олардың жоғарғы мәзірі бірдей, бірақ әдепкі тақырып және контекстік мәзір басқаша.

macOS жүйесінде бір қолданба мәзірі бар. Ол ағымдағы таңдалған терезеге сәйкес динамикалық түрде өзгереді. Оның IDLE мәзірі бар және төмендегі жазбалардың кейбірі Apple ұсыныстарын көрсету үшін жылжытылған.

idle.py [-c command] [-d] [-e] [-h] [-i] [-r file] [-s] [-t title] [-] [arg] ...

-c command run command in the shell window

-d enable debugger and open shell window

-e open editor window

-h print help message with legal combinations and exit

-i open shell window

-r file run file in shell window

-s run $IDLESTARTUP or $PYTHONSTARTUP first, in shell window

-t title set title of shell window

- run stdin in shell (- must be last option before args)

Егер дәлелдер болса:

Егер -, -c немесе r пайдаланылса, барлық аргументтер sys.argv[1:...] ішіне орналастырылады және sys.argv[0] '', '-c' немесе '-r' күйіне орнатылады. Параметрлер диалог терезесінде бұл әдепкі болса да, өңдегіш терезесі ашылмайды.

Әйтпесе, аргументтер өңдеу үшін ашылған файлдар болып табылады және sys.argv IDLE ішіне жіберілген дәлелдерді көрсетеді.