ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ ЖОҒАРЫ БІЛІМ МИНИСТРЛІГІ
«ТОРАЙҒЫРОВ УНИВЕРСИТЕТІ» КеАҚ
FACULTY OF COMPUTER SCIENCE

ТОКЖИГИТОВА Н. К.

ПРОГРАММАЛАУ НЕГІЗДЕРІ БОЙЫНША ЕСЕПТЕРДІ ШЕШУ ПРАКТИКУМЫ

Оқу-әдістемелік құрал

ZHARDEM
SEMEI
2023

ӘОЖ 004.438
КБЖ 16.2 Қаз.
Т 52

Баспаға «Торайғыров университеті» коммерциялық емес акционерлік қоғамы Ғылыми Кеңесі шешімімен ұсынылды
27.12.2023 жылғы №5/4 хаттамасы

Рецензенттер:
Л. Е. Алдибаева – физика-математика ғылымдарының кандидаты, қауымдастырылған профессор, Қазақ ұлттық аграрлық зерттеу университеті;
Л. М. Кыдыралина – PhD, қауымдастырылған профессор м.а., «Шәкәрім атындағы университет»;
Н. Н. Оспанова – педагогика ғылымдарының кандидаты, профессор, «Торайғыров университеті».

Т 52
Токжигитова Н. К.
ПРОГРАММАЛАУ НЕГІЗДЕРІ БОЙЫНША ЕСЕПТЕРДІ ШЕШУ ПРАКТИКУМЫ: [Мәтін:] оқу-әдістемелік құрал. Құраст.: Н. К. Токжигитова. «Toraighyrov University» КеАҚ. Семей: «Zhardem» республикалық баспа компаниясы. – 2023 жыл. – 100 бет.

ISBN 978-601-345-503-7

Оқу-әдістемелік құрал программалау негіздерін оқуда көмекші нұсқаулық болып табылады. Құралдың әр бөлімінде: теоретикалық материалдар, типтік тапсырмаларға арналған шешімдердің мысалдары және материалдарды шоғырландыруға арналған жаттығулар жиынтығы көрсетілген. Қарастырылған барлық мысалдар егжей-тегжейлі түсіндірмелермен, түсініктемелермен және тапсырманы орындау үшін тиісті программалық кодпен бірге жүреді.

Шешілген міндеттер C++ және Visual Studio программалау ортасын игеретін жоғары оқу орындарының білім алуышларына және программалау мәселелеріне қызығушылық танытқандардың барлығына пайдалы болады.

ISBN 978-601-345-503-7

ӘОЖ 004.438
КБЖ 16.2 Қаз.

КІРІСПЕ

Программалау тілі біздің қиялымызды қалыптастырады және нені елестете алатынымызды анықтайды. Дәстүрлі процедуралық программалау моделінде (үлгісінде) программаның орындалуы бірінші жолдан басталады және қажет болған жағдайда әртүрлі процедураларды шақыра отырып, стандартты жолды ұстанады. Объектіге бағытталған тілдерде оқиғаға негізделген программалау моделі (үлгісі) жүзеге асырылады, яғни программа қатаң белгіленген жолмен емес, қандай оқиғалардың орын алатынына байланысты орындалады.

Бұл оқиғалар пайдаланушының әрекеттері немесе жүйенің хабарлары арқылы іске қосылады. Осылайша, орындалатын программаның орындалу реттілігі оқиғалар тізбегі арқылы анықталады. Программаны келесі жолы іске қосқанда, реттілік басқаша болуы мүмкін.

Объектіге бағытталған визуалды программалау жүйелері әзірлеушіге объектілерді құру және олардың қасиеттері мен әдістерін баптау (теңшеу) үшін көптеген құралдарды ұсынады. Объект қасиеттерінің әрқашан атаулары болады, ал баптау (теңшеу) процесін қолмен кодтауға жүгінбестен, арнайы графикалық құралдардың көмегімен жасауға болады. Визуалды програмалау жүйелерінің бұл мүмкіндігі программистің өнімділігін айтарлықтай арттырады және үлкен программаларды жасауды өте қарапайым етеді.

Осы жинақта ұсынылған тапсырмалар тақырыбының реттілігі оларды шешу реттілігіне сәйкес келеді, бұл бағдарламалау тілдерін меңгерушілерге қарапайым және құрылымдық деректердің мүмкін түрлерін, әр түрлі деңгейдегі программаларды құру мен әзірлеу тәсілдерін және тілдегі графикалық объектілерді, сыныптарды, тізім (стек, кезек) типті объектілерді құру, кесте типті объектілер, вектор типті объектілер, ағаш типті объектілер сияқты меншікті объектілер мен сыныптарды жасау жолдарын зерттеуге мүмкіндік береді. Тапсырмалардың күрделілігін бағалау айтарлықтай субъективті және ол, сөзсіз, программалау бойынша білім алушылардың жалпы дайындық деңгейіне байланысты. Осыған байланысты бұл құралда тиісті пәндер бойынша сабақтар өткізу үшін де, немесе өзіндік жұмыс үшін де қолдануға болады. Мұндай есептер жинағын пайдалану көптеген практикалық мәселелерді шешуде әртүрлі деректер құрылымдарын әзірлеу әдістерін игеру үшін орынды.

Оқу - әдістемелік құралда қиындық деңгейі бойынша реттелген тапсырмалар қамтылған.

Жинақ есептің шарттарына және шешу алгоритмдеріне сәйкес құрылымдалған. Программаның орындалу нәтижелері ұсынылған.

Келтірілген алгоритмдердің жалпы және дербес жағдайы берілген. Жалпы жағдай үшін берілген блок-схемалар кез-келген бағдарламалау тілінде программалар жасауға мүмкіндік береді. Алгоритмнің дербес жағдайы Visual Studio ортасында C++ тілінде сипатталған.

Оқу – әдістемелік құралдың бірінші тарауында С ++ тілінің негізгі элементтерін: стандартты деректер түрлері, тұрақты және айнымалылар, ағындық кіріс/шығыс ұйымдастыру, операциялар және өрнектерді қамтиды, практикалық мәселелерді шешу үшін функцияларды жобалау, әзірлеу және қолдану принциптері қарастырылады.

Екінші тарауда есептерді шешу алгоритмдері, программалық кодтарына арналған.

Үшінші тарауда программалау бойынша оқыту нәтижелерін бағалау критерийлері анықталған.

Оқу әдістемелік құралда әрбір бөлімінде: теориялық материал, типтік есептерді шешу мысалдары және материалдарды түзетуге арналған жаттығулар жиынтығы берілген.

Оқу – әдістемелік құралдың тарауларында шешілген есептер жинағымен, блок-схемалармен, С++ және Visual Studio программалау ортасымен танысу болады, онда әрекеттер, операторларды басқару, деректер операциялары сияқты жалпы және нақты програмламалау жағдайларының мысалдары қарастырылады, жалпы жағдайда блок-схемалар кез-келген программалау тілінде программа жасауға мүмкіндік болады

1 C++ ТІЛІНІҢ НЕГІЗГІ ЭЛЕМЕНТТЕРІ

1.1. Бағдарлама құрамы. Бағдарлама алфавиті

Алфавит - бұл тілде рұқсат етілген таңбалар жиынтығы. C++ тілінің алфавиті ретінде кіретіндер:

1) бас әріптер мен кіші әріптер және астын сызу;

2) 0-ден 9-ға дейін араб сандары, A-дан F-дейін он алтылық сандар;

3) арнайы белгілер: " { } . ׀ ; [ ] ( ) + - / % * . ? <=> ! & # ~^

4) кеңістік таңбалары: бос орын, кесте таңбасы, жаңа жол таңбасы.

Алфавиттік таңбалардан тілдің лексемдері анықталады: идентификаторлар, негізгі (резервтік) сөздер, операциялық белгілер, тұрақты, шектегіштер (жақшалар, кезең, үтір, бос орын).

Лексемдердің шекаралары бөлгіштер немесе операция белгілері сияқты басқа лексемалармен анықталады. Өз кезегінде, белгілер өрнектердің бір бөлігі (өрнекте белгілі бір мәнді есептеу ережесін анықтайды) және операторлар (оператор кейбір әрекеттің толық сипаттамасын анықтайды).

Кейбір белгілер, мысалы, орыс алфавитінің әріптері C ++-де пайдаланылмайды, бірақ олар ескертулер мен таңба тұрақтыларына енгізілуі мүмкін.

Идентификаторлар

Идентификатор - бұл бағдарлама нысанының атауы: тұрақты, айнымалы, таңба, тип, сыныптың данасы, функция және жазудағы өріс. Идентификатор латын әріптерін, сандарды және астыңғы сызықтарды қамтуы мүмкін. Үлкен және кіші әріптер түрінде, мысалы, myname, myName және MyName- әр түрлі үш атауда. Идентификатордың бірінші таңбасы әріп немесе астыңғы сызық болуы мүмкін, бірақ сан емес. Аттардағы бос орындарға жол берілмейді.

C ++ атаулардың ұзындығына қандай да бір шектеулер енгізбейді, алайда оқу және жазу кодының ыңғайлылығы үшін оларды тым ұзақ жасамау қажет.

Түйінді сөздер

Түйінді сөздер - компилятор үшін ерекше мағынасы бар сақталған идентификаторлар, мысалы, include, main, int және т.б. Түйінді сөздер тек қана мақсатқа арналған. Түйінді сөздер мен олардың мақсатын C ++ анықтамалық жүйесінде табуға болады.

1.2. Бағдарлама құрылымы

Әр С ++ бағдарламасы бір немесе бірнеше функциялардан тұрады. Әрбір функция төрт негізгі элементті қамтиды:

1) қайтару түрі;

2) функция атауы;

3) жақшаға алынған параметрлер тізімі (бос болуы мүмкін);

4) функция денесі (операторлардың тізбегі), ол бұғана жақшаларда орналастырылған.

Программаны іске қосқан кезде, функция шақырылады main- бұл бағдарламада міндетті түрде қатысуы керек бағдарламаның басты функциясы. Қарапайым функцияның мысалын келтіреміз main():

```
int main()
{return 0;}
```

Келтірілген мысалда:

1) қайтару түрі int-толықтай;

2) функция атауы main;

3) бос параметрлер тізімі

4) функция денесі бір оператордан тұрады return 0, ол қоңырау ортасына 0 ге қайтарады (компилятор немесе операциялық жүйе).

С ++ функциясының ескі нұсқаларында main() функциясын void түрінде көрсетуге болады. Бұл дегеніміз, бұл функция ештеңе қайтармайды. Бұл жағдайда оператор *return* қажет емес. С++ функциясының қазіргі нұсқаларында main() әрдайым int түрін қайтарады, ол бағдарламаның әдеттегі немесе қалыпсыз өшірілуінің қоңырау ортасына нұсқайды. Бірақ функция параметрлерінің тізімі бос болмауы мүмкін, бірақ ол параметрлер санында шектелген.

Енді қарапайым бағдарламаның мысалын мысалға келтіріңіз, оның мәндері пайдаланушының пернетақтадан енгізілген екі бүтін санның қосындысын есептейді және нәтиже көрсетіледі.

Бағдарламаны компьютердің жадына кірер алдында, мамандандырылған әзірлеу ортасында жұмыс істеу Microsoft Visual Studio қағидаттарымен танысыңыз (қосымша 2) және қателермен, бағдарлама кезінде кездессе (қосымша 2) назар аударамыз.

```
#include <iostream>       // препроцессорлық директива
/* мәндері пайдаланушыдан пернетақтадан енгізілген екі бүтін сандарды есептейтін бағдарламаның мысалы және нәтиже экранда көрсетіледі*/
using namespace std;
int main ()
{int a, b;                                   // айнымалылардың сипаттамасы
cout <<" 2 жай сан енгіз " <<endl;      //қорытынды оператор
cin >>a >>b                                 //енгізу оператор
```

cout <<"Қосындысы" <<a+b; //қорытынды оператор
return 0;}

Бірінші жолдағы кодты қарастырамыз: #include <iostream> - бұл компиляторға C ++ бағдарламасының кітапханалық тақырып файлын құрастырмастан бұрын мазмұнды қосуды бұйыратын препроцессор директивасы. Тақырып файлының атауы бұл iostream жағдайда жақшада орналастырылады. Бұл файлда пішімделген енгізі / шығаруды ұйымдастыру үшін дайын бағдарламалық жасақтама бар. Жалпы, бағдарламада бірнеше нұсқаулар болуы мүмкін, олардың әрқайсысы жаңа жолдан бастау керек және функцияның сыртында орналасуы керек.

Кітапхана атауы *iostream* келесі сөздердің қысқаруынан келеді input-output (кіріс-шығыс) stream (ағын). Ағын - кез келген тәсілмен жазылған немесе оқылатын ақпарат енгізу / шығару құрылғысынан оқылатын таңбалар тізбегі. Сондықтан, кітапханада енгізу/шығару ағыны деректерін ұйымдастыруға арналған кітапхана да аталады.

Бұдан кейін *пікір* беріледі. Түсіндірмелер пайдаланылған айнымалылар, алгоритм немесе пішімделген енгізу / шығару туралы қысқаша ескертулер үшін пайдаланылады. Түсіндірмелер пайдаланылған айнымалылар туралы қысқаша ескертулер үшін, алгоритм немесе күрделі код бөлігін әрі қарай түсіндіру үшін пайдаланылады, яғни. бағдарламаның мағынасын түсінуге көмектеседі. Пікірлер компилятормен еленбейді, сондықтан олар компилятордан кодтың бір бөлігін «жасыру» үшін пайдаланылуы мүмкін.

C ++-те түсініктемелердің 2 түрі бар:

1) Мазмұны бір жолға сәйкес келетін түсініктеме таңбалармен басталады // және жаңа сызықтық таңбамен аяқталады.

2) Егер түсініктеме мазмұны бір сызыққа сәйкес болмаса, таңбалар арасында орналасқан жұптық түсініктеме / * */ қолданылады. Ішкі түсініктемеде, кез келген таңбалар, соның ішінде қойынды таңбасы және жаңа жол таңбасы болуы мүмкін. Қосарланған түсініктеме тіркемелерге рұқсат бермейді.

Біздің мысалда екінші және үшінші жолдар - жұпталған пікірлер.

Төртінші жолда директива бар using namespace std, бұл бағдарламада төменде анықталған барлық аттар есім кеңістігіне қатысты болады std. *Атаулар кеңістігі* - белгілі бір аттар жиыны анықталған бағдарлама аймағы. Бұл аттар осы аттар кеңістігінен тыс белгісіз. Біздің жағдайда бұл жаһандық (яғни кітапхананың аттар кеңістігінде анықталған *iostream)* атаулар cout, cin және endl атаулар кеңістігінде std анықталады, сонымен қатар, ол өздерінің атауларын анықтайды, a, b. Атаулар кеңістігі std- бұл кеңістік атауында стандартты кітапхана пайдаланылады. Бағдарламашы аттар кеңістігін анықтауға құқылы.

Бағдарламада орын алған әр бірегей бірегей болуы керек. Үлкен және күрделі қосымшаларда түрлі өндірушілердің кітапханалары пайдаланылады.

Бұл жағдайда оларда қолданылатын атаулардың арасындағы қақтығысуға жол бермеу қиын. Аттар кеңістігі атаулардың қақтығыстарын болдырмаудың қарапайым механизмін қамтамасыз етеді. Олар жаһандық аттар кеңістігінде бөлімдер жасайды, программалық нысандардың ауқымын және олардың атауларының бірегейлігін бөледі.

Біздің бағдарламамыздың келесі жолы - бұл функцияның атауы main. ол параметрлерді қамтымайды, бірақ түрдің мәнін int қайтаруға тиіс. Функция денесі бұйра жақтауларға орналастырылады.

Алтыншы жол екі түрдегі айнымалыларды жариялайды int (мәліметтердің толық түрі). С ++ тілінің көбірек желілік түрлері 1.5 бөлімінде, айнымалы - 1.7 бөлімінде талқыланады.

Жетінші жолы идентификатордан басталады *cout,* бұл пернетақтаға бағдарланған стандартты кіріс ағынымен жұмыс істеуге арналған С ++ нысаны. Әрі қарай операция <<, ол кірістіру немесе енгізу операциясы деп аталады. Бұл әрекет таңбалардың оң жағындағы мазмұнды көшіреді <<, сол таңбалардың сол жағындағы нысанға. Бір жолда ағынға орналастырудың бірнеше операциясы болуы мүмкін. Біздің жағдайда, команданы орындау нәтижесінде cout<< экранда «Екі бүтін сан енгізіңіз» деген жол пайда болады және курсор жаңа жолға ауысады. Курсорды жаңа жолға жылжыту арнайы сөзді *endl* пайдалану арқылы жүзеге асады, манипулятор деп аталады: шығыс ағынына жазылған кезде хабар жаңа жолға аударылады.

Келесі жол *cin* идентификатордан басталады. Идентификатор cin - клавиатураға бағдарланған стандартты кіріс ағынымен жұмыс істеу үшін арналған С ++ нысаны. Әрекет ағыннан экстракция операциясы деп аталады: ол объектінің мәндерін оқиды және оларды таңбалардың оң жағындағы айнымалыларға сақтайды >>. Осы операцияның нәтижесінде пернетақтадан екі мән енгізіледі, олар a айнымалыда сақталады, екіншісі b айнымалыларда сақталады.

Бұдан шығатын қорытындыны айтсақ- мәлімдеме шығады, нәтижесінде экранда хабарлама пайда болады «Олардың суммасы тең ...». Мысалы, пернетақтадан 3 және 4 нөмірлерін енгізсеңіз, экранда «Олардың суммасы 7 тең» деген хабарлама пайда болады.

1) Егер сізде ескі компилятор болса, онда көрініс тақырыбы #include <iostream>ауыстыруға тура келеді #include <iostream.h> сызықты алып тастаңыз using namespace std

2) С, С ++ мұраланған енгізу / шығару функциялары scanf() және print(),олар файлды қосу арқылы пайдалануға болады stdio.h.

1.3. Стандартты С ++ деректер түрлері

Деректер- ақпараттың форматталған (компьютермен түсінікті) ұсынылуы. Бағдарламаларда деректер айнымалы мәндер немесе тұрақты мәндер ретінде көрсетіледі. Бағдарламаны орындау барысында өзгермейтін деректер тұрақты мән деп аталады. Бағдарламада жарияланған және өзгертілген деректер айнымалы деп аталады. Деректерді ұсыну ерекшеліктері:

1) әрбір мән (айнымалы, тұрақты және функциямен қайтарылған нәтиже) өз түріне ие;

2) айнымалы немесе тұрақты мәндері оларды сипаттағанда жарияланады;

3) түрін анықтайды:

- компьютерлік жадыдағы деректердің ішкі көрінісі;
- бұл түрдегі мәнді орналастыру үшін талап етілетін кез келген сан;
- бұл түрдің мәндерін қабылдайтын мәндердің жиынтығы;
- операциялар мен функцияларды, осы типтегі мөлшерде қолдануға болады.

Деректердің барлық түрлері қарапайым және композитті болуы мүмкін. Қарапайым: стандартты (бүтін, нақты, символдық, логикалық) және пайдаланушы анықталған (сандар түрінде). Құрамдастырылған түрлерде массивтер, жолдар, бірлестіктер, құрылымдар, файлдар және нысандар бар. Сонымен қатар, мәндерді сақтауға арналмаған және мәндерді қайтармайтын функцияларды анықтау үшін пайдаланылатын арнайы түрі бар. Бұл бөлімде стандартты деректер түрлерін ғана қарастырамыз.

Барлық деректер түрлері

Толық деректер түрлерінің отбасы ретінде - *short, int, long;* және екі спецификаторлар- *signed* және *unsigned* (кесте 1.1).

Кесте 1.1

Түрлері	Ауқымы	Өлшемі (байт)
short	-32 768... 32 767	2
unsigned short	0... 65 535	2
int	-32 768 ... 32 767 немесе -2 147 648...2 147 483 647	2 және 4
unsigned int	0 ... 65 535 және 0 ... 4 294 967 295	2 және 4
long	-2 147 483 648 ... 2 147 483 647	4
unsigned long	0 ... 4 294 967 295	4

int түрі бұл жабдыққа тәуелді болып табылады, бұл 16 биттік процессор үшін 32 биттік байт үшін бұл мәндер 4 түрі үшін 2 байт бөлінгенін білдіреді. - *short* түрі-2 байта, *long*- 4 байт әрдайым тіркелген.

Іріктеуішті *unsigned* деректер ұсынудың қол қойылмаған пішімін орнату үшін қолданылады, *signed*- бағдар. Әдепкі бойынша, барлық бүтін сан түрлеріне қол қойылады, сондықтан *signed* іріктеуішті алып тастауға болады.

Таңбалар түрлері *char* және *wchar_t* бүтін сандарды сақтау үшін де пайдаланылуы мүмкін, бірақ біз бұл тұрғыда осы түрлерді қарастырмаймыз.

Нақты деректер түрлері

Нақты деректер түрлерінің отбасы ретінде- float, double, long double (кесте 1.2).

Кесте 1.2

Түрлері	**Ауқымы**	**Өлшемі (байт)**	**Ондық белгіден кейін маңызды сандардың саны**
float	3.4e-38...3.4e+38	4	6
double	1.7e-308... 1.7e+308	8	15
long double	3.4e-4932 ...3.4e+4932	10	30

1.2-кестеде ең төменгі және ең жоғары мәндердің абсолюттік мәндері көрсетілген.

Таңбалар түрлері

Таңбалар түрлері char және wchar_t түрлері бойынша сипатталады.

Түрдің мәні 1 байт болып тағайындалады, ол 256 таңбалы жиыннан ASCII кез келген таңбаны орналастыру үшін жеткілікті. wchar_t түрдің өлшемі 2 байт, ол Unicode таңбалар жинағынан кез келген таңбаны орналастыру үшін жеткілікті.

Логикалық таңбалар түрі

Логикалық деректерді сипаттау үшін bool түрі қолданылады, яғни әрдайым екі мәнді білдіретін: true (шындық) және false (жалған). Теориялық түрдегі айнымалы өлшемі bool, 1 бит болуы керек, бірақ іс жүзінде көптеген компиляторлар оған 1 байт бөледі, себебі аппараттық деңгейде байтқа қолжетімділікті ұйымдастыру битке қарағанда оңай.

1.4. Тұрақтылар

Тұрақтылар - бұл бағдарламаны орындау барысында өзгермейтін деректер. С ++ жүйесінде атын және атаусыз тұрақты мәндерін пайдалануға болады.

Нақты тұрақтылар

Аталған тұрақты немесе литералдар әдеттегі тіркелген мәндер болып табылады. Мысалы, операторда:

```
a=b+2.5;          //2.5-бекіткен тұрақты
```

Бүкіл, нақты, символдық және жол мәтіндері бар. Компилятор сыртқы түрі бойынша деректер түрлерінің біріне литерал береді.

Бүтін сан формасын жазу үшін, үш түрдің бірін пайдалана аласыз: ондық, сегіздік немесе он алтылық. Ондық сан: ондық сандар тізбегі, егер ол сан болса,

нөлден басталмайды. Сегіздік: нөл, одан кейінгі сегіздік сандар (0,1,2,3,4,5,6,7). Он алтылық: Ох немесе ОХ болып келетин он алтылық сандар (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

Мысалы, 20 мәнін осылайша жазуға болады:

1) ондық түрінде 20-ға тең;

2) сегіздік түрінде 024;

3) он алтылық түрінде 0x14.

Әдепкі бойынша, бүтін литералдар int немесе long типі бар. Нақты түрі мәнге байланысты. Жиынты қосу арқылы бүтін тұрақты мән түрін анықтай аласыз: long, unsigned, unsigned long. Мысалы, 1L (long түрі) 8Lu (unsigned long), 128u (unsigned).

Нақты әріптерді жазу үшін ондық немесе экспоненталық пішінді пайдалануға болады.

Жазбаның ондық формасы пішінге ие: *[бүтін бөлік], [бөлшек бөлік],* мысалы, 2.02.-10.005. Бұл жағдайда нөмірдің енгізілуінде немесе бүтін бөлік немесе бөлшек бөлік бір уақытта екеуіне де жіберілмеуі мүмкін. Мысалы: 2 және 002.

Экспоненттік нысанда түрлері *[мантиса] {Е/е}[+/]][рет]* бар, мысалы 2.02.-10.005. Бұл жағдайда тұрақтының мәні санның тәртібімен анықталған қуатқа 10 мантисаның өнімі ретінде анықталады. Мысалы, жазбада 0.2E6 эквивалентті жазылуы $0.2 \cdot 10^6$, ал 1.123E-2-эквиваленті $1.123 \cdot 10^{-2}$ ретінде жазылады.

Әдетте, өзгермелі нүктенің тұрақты мәндері double түрі бар. Мәннен кейін бір дәлдікті көрсету үшін, жұрнақ f немесе F, ұлғайтылған жұрнақтар үшін l немесе L

Мысалы:

Ондық жағдайда	Экспоненциальды формада
3.14159F	3.14159E0f
.001f	1E-3F
12.345L	1.2345E1L
0	0e0

Таңбалы әріптер – бұл бір немесе екі таңбалар, апострофтерге салынған. Мысалы, 'f', '. ', '3', /n'. Көптеген таңбалар саны басылатын таңбалар болып табылады, яғни. Олар экранда келтірілген пішінде көрсетіледі. Басып шығарылмайтын таңбаларды енгізу үшін бақылау реті пайдаланылады. Басқару реті солға қарай қиғаш сызықшадан басталады. Кейбір С ++ басқару тізбектері 1.3-кестеде келтірілген.

Кесте 1.3.

Нысаны	Атауы	Нысаны	Атауы
\a	Дауысты сигнал, хабарлама(alert)	\t	Көлденең табуляция (horizontal tab)
\b	1 таңбаға оралыңыз (backspace)	\v	Тік табуляция (vertical tab)

\f	Бетті аудару (formfeed)	\\	Кері қиғаш сызық (backslash)
\n	Жолдарды ауыстыру, жаңа жол (newline)	\'	Апостроф, бір тырнақша(single quote)
\r	Тасымалдауды қайтару (carriage return)	\»	Тырнақша

Жолдың литералы – мысалы, тырнақшадағы таңбалардың ерікті саны. *Алақай! Бүгін инфораматика сабағы!*

Сызықты литералдар, сонымен қатар, 1.3-кестеден бақылау тізбектерін қамтуы мүмкін. Мысалы, жолдың ішінде тырнақшалар жазғыңыз келсе, оларды слайдтан алдын ала қойыңыз, олардың үстіне компилятор оларды жолды бөлетін тырнақшалардан ажыратады: «Дж.Р.Р. Толкин/ «Властелин Колец/» «.

Қос жолдармен және бос орындармен, табуляциялармен немесе жаңа жолдармен орналасқан екі жол мәтіні бір жаңа жолдың литералына біріктіріледі. Бұл бірнеше жеке сызықтары бар ұзын жол мәтіндерін жазуды жеңілдетеді. Ұзын сызықтарды жасаудың тағы бір қарапайым жолы бар: таңбалар жолының соңына қиғаш сызуды солға орналастыру арқылы келесі жолдың осы жолмен бір бүтін сан болатындығын көрсетуге болады. Солға қарай қиғаш сызық сызықта соңғы болуы керек, кейін түсініктемелер мен бос орындар болмауы керек. Сонымен қатар, келесі жолдың басындағы барлық бос орындар мен қойындылар алынған жол ұзындығының бөлігі болады. Бір таңбалық жолдың арасындағы айырмашылықты ескеріңіз, мысалы: «А» және «А» таңбасының тұрақты мәні. Сонымен қатар бос жолдың литералы рұқсат етіледі, бірақ таңба жолы жоқ.

Атаулы тұрақтылар

Егер шешім осы тапсырма үшін маңызы бар тұрақты мәндерді қолданса, оларды программада аталған тұрақты мәндер ретінде анықтай аласыз. Атаулы тұрақтының хабарландыру форматы:

[<жады сыныбы>] <const> <тип> <атаулы тұрақты аты>= <өрнек>

мұндағы *жады сыныбы*- бұл бағдарлама объектісінің қызмет ету мерзімі мен ауқымын анықтайтын (2.4 бөлімді қараңыз) спецификаторлар; *өрнек* аталған тұрақты мәнін анықтайды, яғни оны инициализациялайды.

Тұрақты декларацияда инициализациялау керек. С ++ жүйесінде операндтар бұрын анықталған тұрақты мәндер болуы мүмкін өрнектің тұрақты мәндерін жариялау кезінде пайдалануға рұқсат етіледі. Сондай-ақ, бір мәлімдемені сол түрдегі бірнеше тұрақты мәндерді үтірлермен бөліп көрсете аласыз. Мысалы:

const int I=-124;

const float k1=2.345, k2=1/k1;

1.5. Айнымалылар

Айнамалылар - бұл белгілі бір түрдегі деректер сақталатын аталатын еске сақтау аймағы. Айнымалы мән атауы және мәні бар. Атауы мән сақталатын жад аймағына кіру үшін пайдаланылады. Бағдарламаны орындау кезінде айнымалы мән өзгертілуі мүмкін. Қолданар алдында кез-келген айнымалы сипатталуы керек. Бағдарламаны орындау кезінде айнымалы мән өзгертілуі мүмкін. Қолданар алдында кез-келген айнымалы сипатталуы керек. Айнымалылардың жазылу (хабарландыру) форматы:

[<жады сыныбы>] <тип> <аты> <=өрнек> (<өрнек>)];

Мысалы, айнымалыны сипаттасақ i, j- int типі және x айнымалы типі double:

int i, j;

double x;

Қолданар алдында кез-келген айнымалы мән анықталуы керек. Мұны төмендегідей жасауға болады:

1) Оператор тағайындау:

int a; //айнымалыны сипаттау

....

a=10; //айнымалы мәнін анықтау

2) Операторды енгізуі:

int a; //айнымалыны сипаттау

...

cin >> a; //айнымалы мәнін анықтау

3) Инициализация - сипаттама кезеңінде айнымалы мәнін анықтау. С ++ тілі айнымалы мәндерді инициализациялаудың екі түрін қолдайды: баптандыруды және тікелей инициализацияны көшіру. Мысалы:

int i=100; // инициализация көшіру

int i (100); // тікелей баптандыру

Бір сипаттамада екі немесе одан да көп айнымалы мәндерді инициализациялауға болады, олардың әрқайсысы жеке мән береді. Сонымен қатар, бір сипаттамада инициализацияланған және жаңартылмаған айнымалылар болуы мүмкін. Мысалы:

int i, day=3, year=2007;

Маңызды тағайындаудан немесе кіріс туралы мәлімдемені пайдаланудан басқа, кез келген мақсаттар үшін пайдаланылмаған айнымалыны пайдаланғанда, белгісіз нәтиже алынады. Кездейсоқ мән жадта айнымалы көрсетілетін мекен-жайда сақталады. Сондықтан қарапайым ережені пайдалану қажет: егер айнымалы бірінші әрекет - оны тағайындау керек болса, онда оны инициализациялаудың қажеті жоқ, әйтпесе, оны сипаттамаға мән беру керек.

1.6. Консольді енгізуді / деректерді шығаруды ұйымдастыру

1.1 бөлімінде енгізу/ шығару бұрыннан қарастырылған. Еске салайық, C ++ -те консольдық деректерді енгізу / шығару («қара» терезеде енгізу / шығару) ұйымдастыру үшін тақырып файлы iostream. Бұл файлда анықталғаны:

1) *cin* объектісі, ол стандартты енгізу-пернетақта құрылғысынан кіруге арналған

2) *cout* объектісі, деректерді стандартты шығыс экран құрылғыларына шығару үшін арналған;

3) операция >>, ол кіріс ағыны деректерін шығару үшін пайдаланылады;

4) операция <<, ол деректерді ағымдық ағынға жылжыту үшін пайдаланылады;

5) форматталған енгізу/шығару операциясы.

Шығару ағынында орналастыру үшін пайдаланылатын кейбір әрекеттерді қарастырайық:

1) түрдің мәні ағынға бір таңба ретінде орналастырылады және ағында бір өлшемі бар өріске ие болады;

2) жол ағынға таңбалар тізбегі ретінде орналастырылады және ағындағы өрісті алады, бұл жол жолдың ұзындығына тең;

3)бүтін түрдің мәні ағынға ондық бүтін сан ретінде орналастырылады және санның барлық сандарын орналастыру үшін өлшемі жеткілікті ағындағы өрісті алады, ал теріс сан үшін минус белгісі үшін де; оң сандар ағынға белгісіз орналастырылады;

4) нақты түрінің мәні «ондық нүктеден» кейін ағынға 6 сандың дәлдігімен орналастырылады; санның мәніне байланысты экспоненталық пішінде (шын мәнінде өте кішкентай немесе өте үлкен болуы) немесе ондық түрін алуға болады.

Ағындарды басқаруға манипуляторлар қолданылады. Экрандағы ағынды көрсету кезінде ағын фрагментін жаңа жолға беруге мүмкіндік беретін манипуляторды қарастырдық. Енді нақты деректер түрлерінің пішімін және оларды экранда орналастыруды басқаруға мүмкіндік беретін endl манипуляторларды қарастырыңыз.

Манипуляторларды аргументтермен пайдалану үшін тақырып файлын қосу керек.

Нақты деректер түрлерінің пішімін басқару

Басқарылатын жүзбелі нүктенің үш аспектісі бар: *дәлдік* (экрандағы сандардың саны); *жазбаның нысаны* (ондық немесе экспоненталық); Бүтін сандар болып табылатын өзгермелі нүкте мәндері үшін ондық бөлікті көрсету.

Дәлдік көрсетілетін цифрлардың жалпы санын анықтайды. Айнымалы нүктенің мәнін көрсете отырып, ол ағымдағы дәлдікке дейін дөңгелектеніп, қысқартылмаған. setprecision манипуляторды пайдаланып дәлдікті өзгерте аласыз. Бұл манипулятордың аргументтері дәлме-дәлдік болып табылады.

Келесі мысалды қарастырайық:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
```

Жұмыс бағдарламасының нәтижесі:

```
{double i=12345.6789;
    cout <<setrecision(3) <<i <<endl;          1.23e+004
    cout <<setprecision(6)<<i<<endl;           12345.7
    cout <<setprecision(9) <<i <<endl;         12.345.6789
    return 0:}
```

Өзгермелі нүктелік мәннің нысаны оның өлшеміне байланысты болады: егер сан өте үлкен немесе өте кіші болса, экспоненталық пішімде, әйтпесе ондық форматта көрсетіледі. Шығу пішімін өз бетінше орнату үшін манипуляторларды scientific (экспоненциалды пішімде қалқымалы нүктелерді көрсету) немесе fixed (ондық форматта өзгермелі нүктелерді көрсету) қолдануға болады.

Келесі мысалды қарастырайық:

```
include <iostream>
using namespace std;
int main()
```

Жұмыс бағдарламасының нәтижесі

```
{double i=12345.678
 cout <<scientific  <<i <<endl;          1.234568e+004
cout << fixed <<i <<endl;                12345.678900
return 0:}
```

Ондық бөлшек болса, әдепкі бойынша көрсетілмейді. Манипулятор showpoint ондық бөлікті мәжбүрлеп көрсетуге мүмкіндік береді.

```
#include <iostream>
using namespace std;
int main()
```

Жұмыс бағдарламасының нәтижесі

```
{double i=10;  cout << i  << endl;          10
cout << showpoint<<i <<endl;                10.0000
return 0:}
```

Экрандағы деректерді орналастыруды басқару

Көрсетілген деректерді орналастыру үшін манипуляторлар қолданылады

1) *left*- шығысын солға теңестіреді

2) *right*- шығысын оңға теңестіреді

3) *internal* - теріс мәнді орналастыруды басқарады: таңбаны солға және оң жаққа қарай теңестіреді, олардың арасындағы бос орындарды босатады.

4) *setprecision (int w)*- белгіленген санмен (манипулятор fixed) форматта нақты сандар үшін бөлшек бөліктегі сандардың максималды санын немесе экспоненталық форматта сандар үшін маңызды сандардың жалпы санын белгілейді (манипулятор scientific)

5) setw (int w)- шығыс өрісінің ең үлкен енін орнатады.

Берілген манипуляторларды мысалға келтіретін болсақ.

#include <iostream>
#include <iomanip>
using namespace std;
int main()

Жұмыс бағдарламасының нәтижесі

{cout <<"1." <<setw(10)<<-23.4567<<endl;
1. -23.4567
cout <<"2." <<setw(10) <<setprecision(3) <<-23.4567 << endl;
2. -23.5
cout <<"3." <<setw(10) <<internal <<-23.4567 <<endl;
3. -23.5
return 0;}

Манипулятор setprecision параметрлері барлық кейінгі шығыс мәлімдемелеріне де қолданылады.

1.7. Операциялар

С++ операцияларының толық тізбесі олардың басымдықтарына сәйкес (басымдықтардың кему тәртібімен, әр түрлі басымдықтармен жұмыс сызықпен бөлінеді). Осы бөлімде операциялардың бір бөлігін егжей-тегжейлі қарайтын боламыз, ал қалған операциялар қажет болған жағдайда енгізіледі.

Операцияларды бір операнд бойынша біртұтас әрекет ететін операндтардың саны, екі операнда екілік-әрекет етуі, үш операнд бойынша әрекет ететін сандар бойынша жіктелуі мүмкін. Кейбір таңбалар қосарлы және екілік операцияларды белгілеу үшін пайдаланылады. Мысалы, * таңбасы орналасудың біртұтас жұмысын белгілеу үшін де, көбейтудің екілік әрекетін белгілеу үшін де қолданылады. Бұл символ бір немесе екі жақты операцияны білдіреді, ол қолданылатын контексте анықталады.

Унарлы амал

1-ге дейін ұлғайту және азайту операциясы (+ + және - -)

Бұл операциялар, тиісінше, инкремент және азаю деп аталады. Операнға дейін операция жазылған кезде жазба-*префикстің* екі түрі бар, ал. операция операнд алдында жазылған кезде және *постфиксті* операциясы операндтен

кейін жазылады. Префикс инкремент (декремент) операциясы оның операндін көбейтеді (азайтады) және нәтиже ретінде өзгертілген мәнді қайтарады. Постфикстің пропорционалды және азайтылған нұсқалары операндтың бастапқы мәнін қайтарады, содан кейін оны өзгертеді.

Осы әрекеттерді мысал арқылы қарастырайық

```
#include <iostream>
using namespace std;
int main()
{ int x=3, y=4;
cout <<++ x <<"\t" <<--y<<endl;
```

Жұмыс бағдарламасының нәтижесі

```
cout <<x++ <<"\t" <<--y<<endl; 4 3
cout <<x <<\t" <<--y<<endl; 4 3
return 0;}                    5 2
```

Префикс нұсқасы едәуір аз әрекетті талап етеді: айнымалының мәнін өзгертеді және нәтижені бір айнымалы мәнге сақтайды. Постфикс операциясы бастапқы мәнді бөлек сақтау керек, содан кейін оны нәтиже ретінде қайтарады. Күрделі түрлері үшін мұндай қосымша әрекеттер уақытты қажет етеді. Сондықтан постфикстің нысаны қажет кезде ғана пайдалануға мағыналы болады. Күрделі түрлері үшін мұндай қосымша әрекеттер уақытты қажет етеді. Сондықтан постфикстің нысаны қажет кезде ғана пайдалануға мағыналы болады.

Анықталған өлшем sizeof операциясы

Бұл әрекет өрнектің өлшемін немесе байт түрін есептеуге арналған және екі нысаны бар: *sizeof <өрнек>, sizeof (<тур>).* Әрекет өрнекте қолданылған кезде, өрнекті бағалаудың нәтижесі *sizeof* түрін қайтарады. Түрге қолданылғанда, көрсетілген түрдің өлшемі қайтарылады.

Бұл әрекетті мысалға келтіріңіз

```
#include <iostream>
using namespace std;
int main()
cout << sizeof (int)<<endl;
```

Жұмыс бағдарламасының нәтижесі

```
cout <<sizeof (x*10)<<endl:
cout <<sizeof (x*0.1)<<endl;
return 0; }
```

Соңғы нәтиже, нақты тұрақты мәндердің әдепкі мәні бойынша анықталады, оған ұзынырақ түрдегі барлық өрнектің түрі беріледі. Бекіткіштер білдіру үшін қажет. оларда тұрып, бұрын қосылатыннан гөрі артық басымдыққа ие типті кастинг жұмысы есептелді.

Бас тарту операциялары

Арифметикалық бас тарту (немесе бірлік минус -) бүтін санның немесе нақты түрдегі операнд белгісін керісінше өзгертеді.

Ескерту. С ++ -де 0 тұрақты false (*f* жалған) мәні ретінде көрсетіледі және тұрақты true (шындық) мәнінің орнына шығады. Бұл логикалық өрнекте кез келген түрдегі нөлдік мән, оның ішінде нөлдік көрсеткіш, логикалық константаға *false*, ал кез-келген нөлдік емес мәнге *true* логикалық тұрақты ие.

Логикалық терістеу (!) мәнді (жалған), операнд нөлден (шын) болса және операнд нөл (жалған) болса, 1 (шын) мәніне әкеледі. Операнның түрі логикалық, бүтін сан немесе көрсеткіш болуы мүмкін. Осы әрекеттерді мысалмен қарастырсақ:

```
#include <iostream>
using namespace std;
int main()
{int x=3, y=0;
bool f=false, v=true;
cout <<++ x <<"\t" <<--y<<endl;
```

Жұмыс бағдарламасының нәтижесі

```
cout <<-y <<"\t" <<!y<<endl;                    -3 0
cout <<f <<"\t" <<!f<<endl;                      0 1
cout <<v<<"\t" <<!v<<endl;                       0 1
return 0; }                                     1 0
```

Екілік операциялар

Арифметикалық операциялар - барлық арифметикалық түрлерге (мысалы, бүтін және нақты түрлерге), сондай-ақ арифметикалық түрлерге (мысалы, таңба түрлері) түрлендіруге болатын кез келген басқа түрлерге қолданылатын операциялар. Екілік арифметикалық операциялар (басымдықты азайту тәртібінде):

1) көбейту (*), бөлу(/), қалдық бөлшек (%);

2) қосылу (+) және азайту (-).

Бөлу операциясы арифметикалық түрдегі операндаларға қолданылады. Дегенмен, егер екі операнды да бүтін сан болса, онда бөлімнің нәтижесінің түрі, әйтпесе, нәтиженің түрі трансформация ережелерімен анықталады. Бөлімнің қалған бөлігі тек бүтін операндаларға ғана қолданылады. Нәтиженің белгісі іске асыруға байланысты.

Мысалы, бөлу жұмыстары және бөлімнің қалған бөлігін қарастырайық.

```
#include <iostream>
using namespace std;
int main()
```

Жұмыс бағдарламасының нәтижесі

```
{ cout <<100/24 <<"\t" <<100/24.0<<endl;          4       4.7619
```

```
cout <<100/21<<"\t" <<100.0/24<<endl;        4       4.1667
cout <<21%3 <<"\t" <<21%.6<<"\t" <<-21%8<<endl;  0    3   -5
return 0; }
```

Байланысты операциялар (<, <=, >, > =, ==, ! =)

Байланысты операциялар <, <=, >, > =, ==, ! = (аз, аз немесе тең, тең емес, артық немесе тең емес, бірдей емес) - бұл бірінші операнды екінші секілді салыстыратын стандартты операциялар. Операнттар арифметикалық немесе көрсеткішті болуы мүмкін. Операцияның нәтижесі - бұл мән true (шын) немесе false (жалған) (нөлге тең емес кез келген мән шын деп, нөлге тең сияқты). Теңдік және теңсіздік үшін салыстыру операциялары басқа салыстыру операцияларына қарағанда төменірек басымдыққа ие.

Логикалық операциялар (&& және \ ᴗ

Логикалық операция ЖӘНЕ (&&) шын мәнін қайтарады, тек егер екі операндтар да шын болса, әйтпесе операция жалған қайтарады. Логикалық амал НЕМЕСЕ (ıı) шынымен қайтарылады, егер тек кем дегенде бір операнд шын болса, әйтпесе операция жалған болады. Логикалық операциялар солдан оңға қарай орындалады, (&&) операцияның басымдықтары (ıı)операцияның басымдығынан жоғары. Егер бірінші операндтың мәндері бүкіл операцияның нәтижесін анықтау үшін жеткілікті болса, екінші операнд бағаланбайды.

Мысалы, осы операцияларды қарастырайық.

```
 #include <iostream>
using namespace std;
int main()
```

Жұмыс бағдарламасының нәтижесі

```
{ cout <<x\t y \t  && \t  //"<<endl;                x   y    &&    //
cout <<0\t  0\t "<< (0 && 0)<<"\t"<<(0 //0) <<endl;      0   0    0      0
cout <<0\t  1\t "<< (0 && 1)<<"\t"<<(0 //1) <<endl;      0   1    0      1
cout <<1\t  0\t "<< (1 && 0)<<"\t"<<(1 //0) <<endl;          1   0    0
1
cout <<1\t  1\t "<< (1 && 1)<<"\t"<<(1 //1) <<endl;          1   1    1
1
return 0; }
```

Шын мәнінде, ақиқат кестесі логикалық ЖӘНЕ - НЕМЕСЕ операциялары үшін жасалған.

Басымдықты азайту тәртібіндегі операцияларды қарастырамыз. Енді бұл тәртіп бұзылады, себебі келесі белгіде шартты операция болуы керек, ол үш жақты операция болып табылады және біз екілік операцияларды қарастырып

біткен жоқпыз. Сондықтан, алдымен, екілік тапсырмаларды қарастырамыз, содан кейін үш жақты тапсырманы орындаңыз. Егер сіз операциялардың басымдықтарын нақтылауыңыз қажет болса, 2-қосымшаға сілтеме жасай аласыз.

Операцияны орындау (=, *=, /=, %=, + =, - =)

Қарапайым түрдегі операция таңбасы = белгісімен анықталады. Орындалған *операцияның қарапайым* форматы (=):

операнд_2=операнд_1:

Операцияның нәтижесінде операнд_1 мәні есептеледі және нәтиже операнд_2 жазылады. Осындай тізбектерді жазу арқылы бірнеше тағайындау операторларын біріктіруге болады: a=b=c=100. Бұл түрді тағайындау солдан оңға қарай орындалады: орындау c = 100 нәтижесі - айнымалы a адресіне 100-ге тең.

Қарапайым *тапсырма операциясынан* басқа, тапсырманы (+ =) бөлумен бөлу (+ =), бөлу арқылы бөлу (= =), бөліну бөлімі (= =), бөлу (/ =) бөлу, =) және т.б. (3-қосымшаны қараңыз).

Кешенді тағайындау кезінде, мысалы, тапсырмамен бірге, операнд_1 операнд_2-ге қосылады және нәтиже операнд_2-ге жазылады. Яғни c + = a өрнегі c = c + a өрнегінің ықшам жазбасы. Сонымен қатар, кешенді тағайындау операциялары сізге неғұрлым тиімді кодты жасауға мүмкіндік береді, себебі қарапайым тапсырма операциясында оң операндағының мәнін сақтау үшін уақытша айнымалы және күрделі тағайындау операцияларында дұрыс операндтың мәні дереу жазылады.

Үштік операция

Үштік операция (? :)

Үштік операция форматы:

операнд_1 ? операнд_2 : операнд_3

Операнд_1- бұл логикалық немесе арифметикалық өрнек. Тұрақтыларға true (шын) және false (жалған) эквиваленттігі тұрғысынан бағаланады. Егер операнд_1 есептеу нәтижесі шын болса, шартты операция нәтижесі операнд_2 мәнін білдіреді, әйтпесе операнд_3. Бұл әрдайым әрдайым операнд_2 немесе операнд_3. Олардың түрлерін оңай ажырата аламыз. Шартты операция if кейінірек талқыланатын шартты мәлімдеменің қысқартылған түрі болып табылады.

Үштік операция мысалға келтірсек:

```
#include <iostream>
using namespace std;
int main()
{ int x, y, max;
  cin >>x>>y;
  (x>y) ? cout <<x: cout<<y<<endl;      //1
```

Жұмыс бағдарламасының нәтижесі

```
max=(x>y)? x : y;                    //2      x=11 және y=9 үшін
cout<<max<<endl;                             11
return 0;}                                   11
```

1 және 2-жолда бірдей мәселені шешетінін ескеріңіз: екі бүтін санның ең үлкен мәнін табу. Бірақ 2-*жолда*, шартқа (x>y) байланысты шартты амал max айнымалы x мәнге немесе y мәнге жазады. Содан кейін айнымалы мәннің max мәнін бірнеше рет қолдануға болады. 1-жолда ең үлкен мән экранда жай ғана көрсетіледі, ал болашақта бұл мән пайдаланылмайды ол ешқандай айнымалыда сақталмады.

1.8. Өрнек және түрді өзгерту

Өрнек - бұл кейбір мәннің есептелу жолын анықтайтын синтаксистік тіл бірлігі. Өрнек операндалардан, операциялардан және жақшадан тұрады. Әрбір операнд өз кезегінде өрнек немесе оның нақты жағдайлардың бірі - тұрақты, айнымалы немесе функциялар.

Арифметикалық өрнектерде қолдануға болатын тақырып файлының math (cmath) математикалық функциялары 4-қосымшада келтірілген.

Өрнектерде мысал:

(a+0,12)/6 x && y || !z (t*sin(x)-!.05e4)/((2*k+2)*(2*k+3)))

Операциялар басымдықтарға сәйкес орындалады (3-қосымшаны қараңыз). Жақшалар операциялардың тәртібін өзгерту үшін пайдаланылады. Бір өрнекте бірнеше өрнектер жазылған болса, жақшалар қолданылады. Егер бір басымдықтағы бірнеше операциялар бір өрнекте жазылса, біртұтас операциялар, шартты операциялар және орналасу операциялары *оңнан солға*, қалғандары *солдан оңға қарай* орындалады. Мысалы:

a=b=c болған кезде *a=(b=c)*

a+b+c болған кезде *(a+b)+c*

Ішкі процесстерді өрнектің ішінде есептеу тәртібі анықталмады: мысалы, сөйлемдегі (sin(x+2)+cos(y+1)) синусқа қоңырау косинусқа қарағанда ертерек орындалады x+2 деп есептелмейді және ол одан y+1 бұрын есептелетін болады.

Өрнекті бағалау нәтижесі оның мәні мен түрімен сипатталады. Мысалы, егер a және b бүтін түрдегі айнымалылар болса және осылайша сипатталады:

int a=2, b=5.

демек, өрнек a+b дегеніміз 7 және int түрі.

Өрнек әртүрлі операндаларды қамтуы мүмкін. Егер операндтер бірдей болса, операцияның нәтижесі бірдей болады. Егер операндалар әртүрлі болса, онда есептеуді басталар алдында, маңыздылығы мен дәлдігін сақтау үшін қысқа түрлердің ұзағырақ түрлендіріледі. Деректер түрінің иерархиясы 1.4-кестеде көрсетілген.

Кесте 1.4.

Мәліметтер түрі	**Өтілі**
Long double	жоғарғы
Double	
Float	
Long	
int	
Short	
char	төменгі

Өрнектердегі түрді түрлендіру сөзбе-сөз (бағдарламашылардың қатысуынсыз) келесідей: егер олардың операндалары әртүрлі болса, «төменгі» түрдегі операнд автоматты түрде жоғары «жоғарғы түріне» айналады. bool арифметикалық өрнектердің түрі int түрге түрлендіріледі, ал тұрақты true 1, false-0 өзгереді .

Тұтастай алғанда, дәлме-дәлдіктің жоғалуы немесе нақтылық жоғалмай жабық өзгеруі мүмкін. Ауқымды емес мысалдарды қарастырсақ:

```
#include <iostream>
using namespace std;
int main()
{ int a=100, b; float c=4.5, d;
  d=a/c;        //1- дәлдік жоғалтпай
  cout <<"d="<<d<<endl;

Жұмыс бағдарламасының нәтижесі
  b=a/c;        //2- дәлдігі жоғалған              d=22.2222
  cout <<"b=" <<b<<endl; return 0;}                 b=22
```

1 жолда айнымалы a (int түрі) айнымалы c (float типті) болып бөлінеді. Түр иерархиясына сәйкес, бөлу операциясының нәтижесі float түрге түрлендіріледі. Нәтиже мәні айнымалы d (float типке) жазылады. Осылайша, 1 қатарда «төменгі» түрден «жоғары» түрге бір түрлендіру жүргізілді және дәл нәтиже алдық.

2 жолдың нәтижесінде бөліну операциясы да оның float түрі болады. Дегенмен, программист мұндай түрленулерде *өздігінен түрлендіру* операцияларын қолдана алады (3-қосымшаны қараңыз). Бұл әрекеттерді көрсету үшін шағын мысалды қарастырыңыз:

```
#include <iostream>
using namespace std;
int main()
{ int a=150000000;
  a=(a*10)/10;              //1-нәтиже дұрыс емес
```

```
cout <<"a="<<a<<endl;
int b=150000000;
b=(static_cast<double>(b)*10)/10; //2-нәтиже дұрыс
```

Жұмыс бағдарламасының нәтижесі

```
cout <<"b="<<b<<endl;                    a=211509811
return 0; }                              b=150000000
```

1 сызықта біз айнымалыны 10-ға көбейтіп, 15000000-ге тең аяқтаймыз, ол арқылы usigned int тіпті сақталмайды. Бұл жағдайда есептеудің дәлдігін жоғалтуға әкеліп соқтыратын түрлендірілмеген түрдегі конверсия орындалған. 2-жолда, айнымалыны b 10-ға көбейте бастағанға дейін, double айнымалы мәннің түріне нақты түрлендіру. Нәтиже мәні уақытша айнымалыға сақталды. Содан кейін, уақыт айнымалысы double 10-ға көбейтіледі, ал нәтиже түрі үшін жарамды мәндер диапазонында 15000000-ға дейін төмендейді, сондықтан толып кету болмайды.

Ашық типтегі түрленулерді тек оның қажеттілігіне толықтай сенімді болсаңыз және оның не екенін түсінуіңіз керек. Өйткені нақты түрлендіру жағдайында, компилятор деректер түрлерін өзгерткен кезде әрекет дұрыстығын тексере алмайды, қателердің ықтималдығы артады.

1.9. Ең қарапайым бағдарламалардың мысалдары

1. X мәнінің белгілі бір мәні үшін өрнектің мәнін есептейтін бағдарлама жасаңыз $\frac{x^2+\sin(x+1)}{25}$.

Мәселені шешу туралы нұсқау. Бағдарламаны жасамас бұрын, C ++ ережелеріне сәйкес операциялардың басымдықтарын есепке ала отырып, бұл өрнекті қайта жазамыз: (pow(x, 2)+sin(x+1))/25.

```
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;
int main()
{int x; double y;
 cin >>x;                                x=10 i
```

Жұмыс кезіндегі нәтижесі

```
 y=(pow(x, 2)+sin(x+1))/25;    //1              y=3.96
 cout <<"y="<<setprecision(5) <<y<<endl;
 return 0;}
```

Компилятордың кейбір нұсқаларында pow және sin функциялар нақты сандарды өңдей алады. Бұл жағдайда 1-жолды келесідей өзгерту қажет болады (pow(x, 2.0)+sin(x+1.0))/25.

2. Периметрі бар квадраттың аумағын р есептейтін бағдарламаны жазыңыз.

Мәселені шешу нұсқаулары. Бағдарламаны құрастырмас бұрын: математикалық ойларды жүзеге асырамыз. Содан кейін жағы бар квадрат болсын:

р=4а периметрі мына формула бойынша есептеледі: ⇨ ⇨a=$\frac{p}{4}$ $\frac{p}{4}$=$\sqrt{s}$

s=a 2 алаңы мынадай формула бойынша есептеледі a=$\sqrt{s}$

```
#include <iostream>
#include <math.h>
using namespace.std;
int main()
{float p,s;
  cout << «Шаршының периметрін енгізіңіз»; cin >>p;
s=pow(p/4, 2);                                    p=20
cout <<'Шаршының квадраты = "<< s;          шаршы аумағының берілгені 25
return 0;)
```

3. Анықтау болып табылады бүтін сан адал.

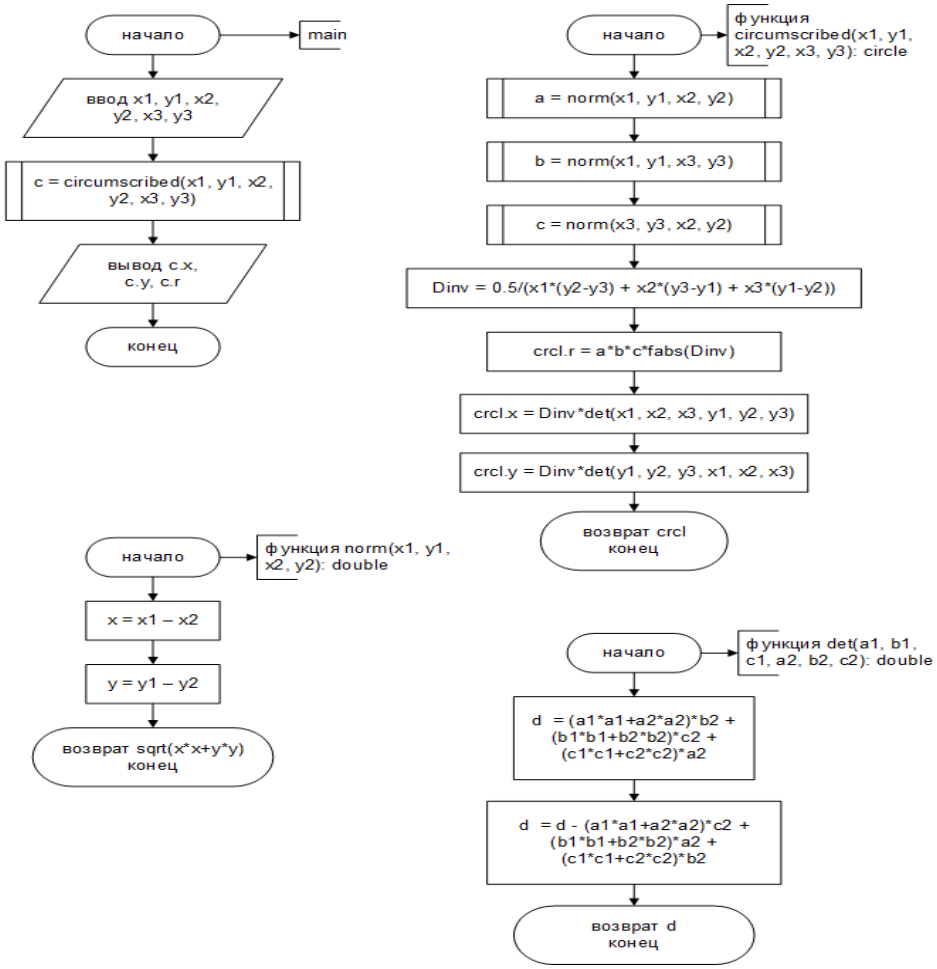
Нұсқау бойынша шешу. Естеріңізге сала кетейік, саны жұп болып табылады, егер қалдық бөлу осы айдың 2 нөлге тең.

```
#include <iostream>
#include <math.h>
using namespace.std;
int main()
{int x;
  cout <<" x енгізу", cin>>x,
 (x % 2=0)? cout <<" тақ/n": cout <<"жұп/n";  //1
Return 0; }
```

1-жолда операция % табады қалдығы деп 2 бөлу. Егер сан жұп болса, онда қалдық 0-ге тең болады, ал С++ нөлдік мәні ретінде түсіндіріледі өтірік. Егер сан тақ болса, онда қалдығы тең болады 1, авС++ ненулевое мәні ретінде түсіндіріледі ақиқат. Сондықтан 1 жолда болады, онсыз салыстыру операциялары. Бұл жағдайда шартты операция мынадай түрде болады: (x%2)?cout <<"жұп/n"; cout <<"тақ/n".

2 ЕСЕПТЕРДІ ШЕШУ АЛГОРИТМДЕРІ, ПРОГРАММАЛЫҚ КОДТАР

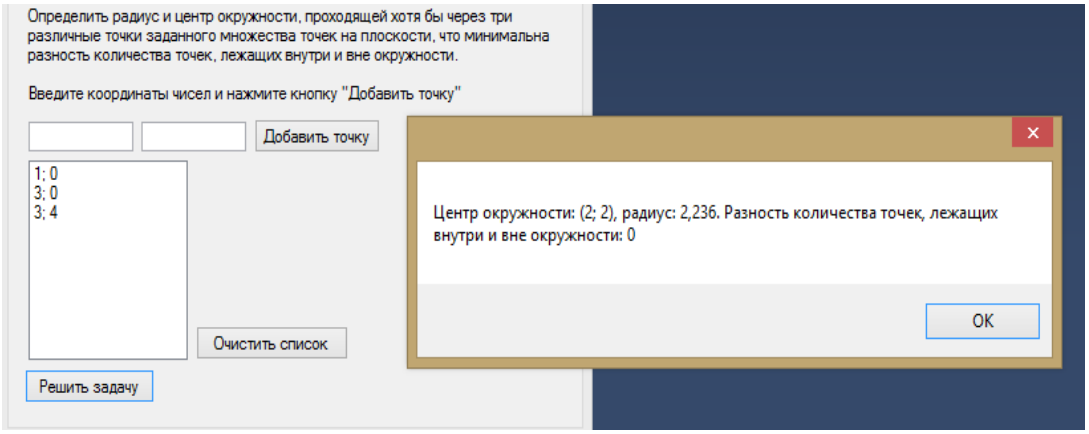
1 Жазықтықта берілген нүктелер жиынының кем дегенде үш түрлі нүктесі арқылы өтетін шеңбердің радиусы мен центрін анықтаңыз, және шеңбердің ішінде және сыртында жатқан нүктелер санының айырмашылығы минималды болуы керек.



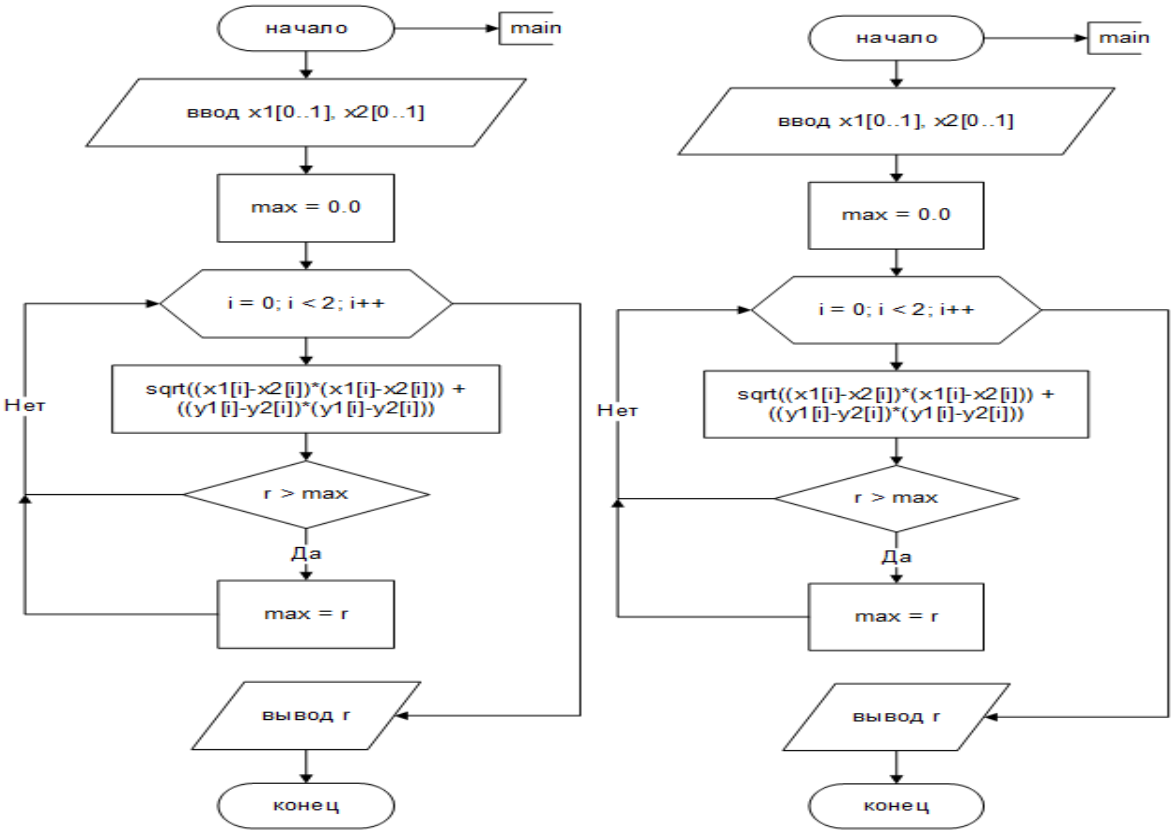

```
#include <stdio.h> // шығару кітапханасы
#include <math.h>   // қосымша математикалық функциялар кітапханасы
#include <iostream> // шығару кітапханасы

// typedef - «circle» сөзі «CIRCLE» құрылымын білдіретін болады. Яғни, оны бас әріппен де, кіші әріппен де жазуға болады (бірақ аралас емес)
// Circle құрылымы - шеңбер үш санмен сипатталады
typedef struct CIRCLE
{
   double x; // шеңбер центрінің x координатасы
```

```
    double y; // шеңбер центрінің y координатасы
    double r; // радиус
} circle;

// кілттік сөз inline - компиляцияны жылдамдатады, бірақ оны өткізіп жіберуге болады
// функция det – алты аргументтердің көмекші функциялары, circumscribed функциясын қара
inline
static double det (
    double a1, double b1, double c1,
    double a2, double b2, double c2)
{
    double d = (a1*a1+a2*a2)*b2 + (b1*b1+b2*b2)*c2 + (c1*c1+c2*c2)*a2;
    d -= (a1*a1+a2*a2)*c2 + (b1*b1+b2*b2)*a2 + (c1*c1+c2*c2)*b2;
    return d;
}

// функция norm – (x1, y1) және (x2, y2) екі нүктенің арасындағы қашықтықты қайтарады
inline
static double norm(double x1, double y1, double x2, double y2)
{
    double x = x1 - x2;
    double y = y1 - y2;
    return sqrt(x*x+y*y);   // Пифагор теоремасы
}

// circumscribed функциясы (x1, y1), (x2, y2), (x3, y3) үш нүктесі арқылы өтетін шеңберді анықтайды, circle объектісін қайтарады (яғни центр координаттары мен радиусы)
circle circumscribed(
    double x1, double y1,
    double x2, double y2,
    double x3, double y3)
{
    circle crcl;
    double a = norm(x1, y1, x2, y2);
    double b = norm(x1, y1, x3, y3);
    double c = norm(x3, y3, x2, y2);
```

```
    // математикалық формулаларды пайдалана отырып, қажетті сандарды табамыз
    double Dinv = 0.5/(x1*(y2-y3) + x2*(y3-y1) + x3*(y1-y2));
    crcl.r = a*b*c*fabs(Dinv);
    crcl.x = Dinv*det(x1, x2, x3, y1, y2, y3);
    crcl.y = -Dinv*det(y1, y2, y3, x1, x2, x3);
    return crcl;
}

int main()
{
    // үш нүкте үшін үш жұп координатаны және бос circle объектісін жариялаймыз  double x1, y1;
    double x2, y2;
    double x3, y3;
    circle c;

    // нүкте координаттарын енгізу
    printf("x1 y1 x2 y2 x3 y3 engiziniz: ");
    scanf("%lf%lf%lf%lf%lf%lf", &x1, &y1, &x2, &y2, &x3, &y3);

    // координаталары берілген үш нүкте арқылы өтетін шеңберді c айнымалысына жазамыз
    c = circumscribed(x1, y1, x2, y2, x3, y3);
    // нәтижені шығарамыз
    printf("Center: (%g, %g)\n", c.x, c.y);
    printf("Radius: %g\n", c.r);

    // Enter пернесін басқаннан кейін консоль жабылады  system("pause");
    return 0;
}
```



2 Жазықтықтағы нүктелер жиынында ара қашықтығы ең максималды болатын нүктелердің жұбын табыңыз.

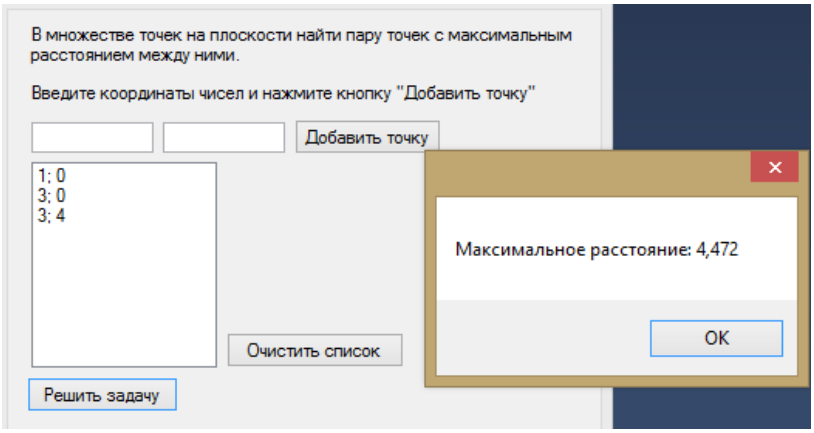



```
#include <iostream>
#include <math.h>
using namespace std;    // барлық жерде std::cin и std::cout жазбау үшін
int main()
{
```

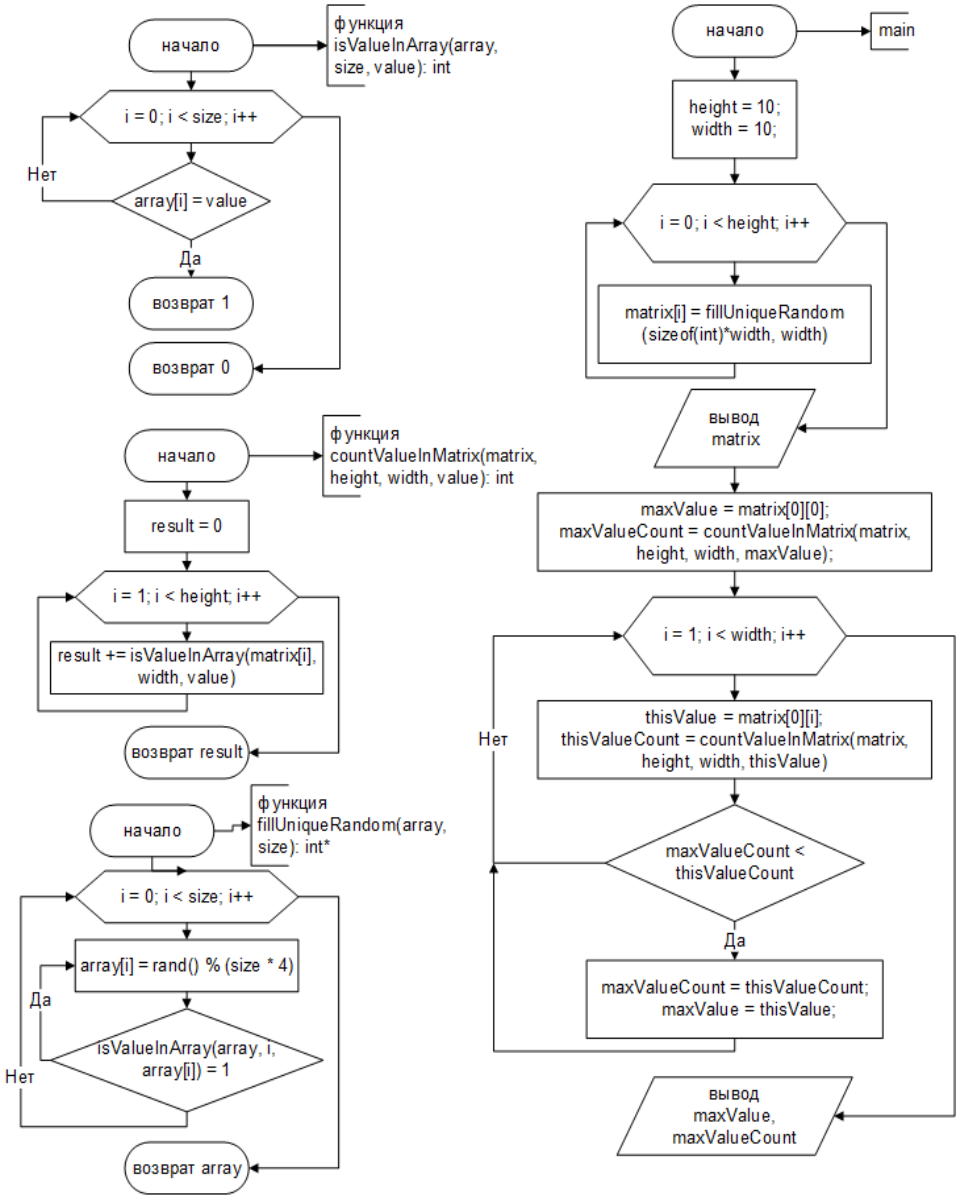
```
// берілген нүктелердің  координаттар жұбы үшін айнымалылар
int x1[2];
int y1[2];
int x2[2];
int y2[2];
// координаттарды енгізу
for (int i=0; i<2; i++){
   cout<<"x"<<i<<": ";
   cin>>x1[i];
   cout<<"y"<<i<<": ";
   cin>>y1[i];
}
for (int j=0; j<2; j++){
   cout<<"x" << j<<": ";
   cin>>x2[j];
   cout<<"y" << j<<": ";
   cin>>y2[j];
}
```

қашықтықтарды табамыз, сонымен бірге нүктелер жұбының арасындағы максималды қашықтықты іздейміз

```
double max=0.0, r;
for (int i=0; i<2; i++)
{
   r=(double)sqrt((x1[i]-x2[i])*(x1[i]-x2[i]))+((y1[i]-y2[i])*(y1[i]-y2[i]));
   if (r>max)
      max=r;  // егер үлкен қашықтықты тапсақ, оны max айнымалысына жазамыз
}
cout<<max;  // нәтижені шығару
   system("pause");
return 0;
}
```



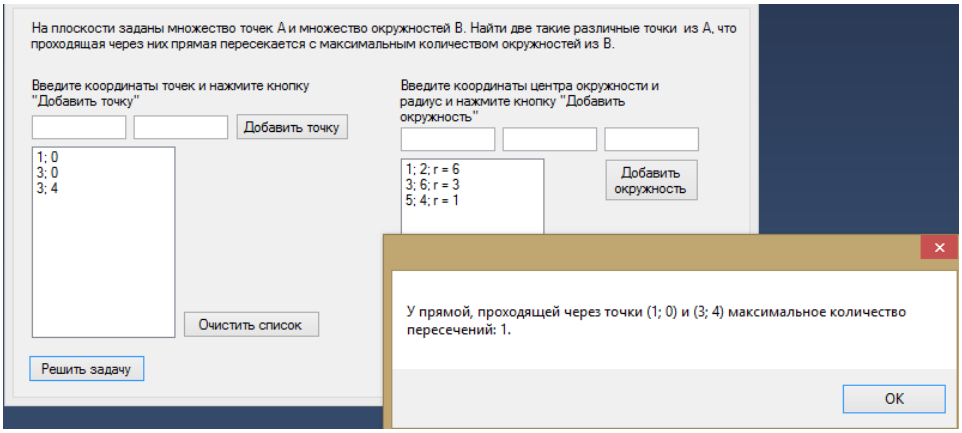
3 Жазықтықта әрқайсысы m нүктеден тұратын n жиын берілген. Бірінші жиынның нүктелерінің ішінен жиындардың ең көп санына жататын біреуін табыңыз.



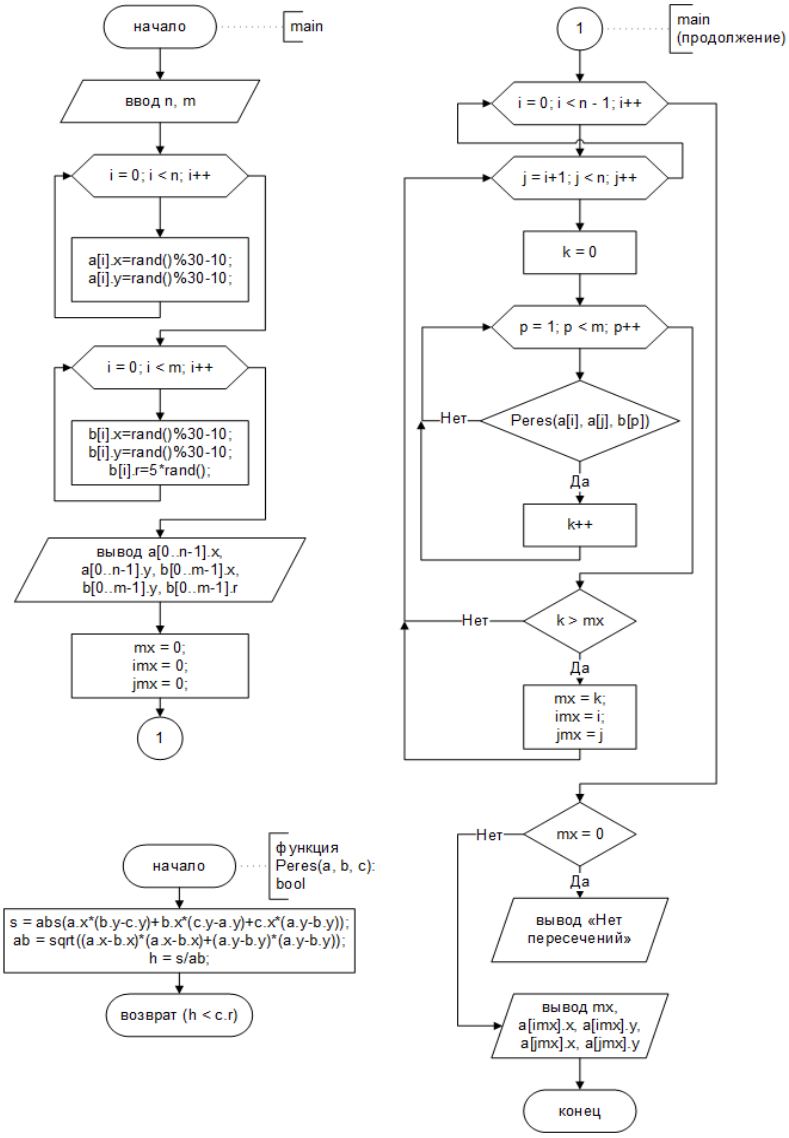
```
#include <stdio.h>
#include <stdlib.h> // кездейсоқ сандар генераторына арналған кітапхана
#include <time.h>   // жүйелік уақытпен жұмыс істеуге арналған кітапхана (кездейсоқ сандар санауышында пайдаланамыз)
#include <iostream>
#include <string>   // жолдармен жұмыс істеуге арналған кітапхана
#include <conio.h> // консольмен жұмыс істеуге арналған кітапхана
 // isValueInArray функциясы - value элементі size ұзындығының array тізімінде болса, 1 қайтарады; әйтпесе 0 қайтарады
int isValueInArray(int *array, int size, int value) {
  int i;
  for (i = 0; i < size; ++i)
    if (array[i] == value) return 1;
  return 0;
}

// массивті әртүрлі сандармен толтыру, бастапқы тізімді қайтарады
int *fillUniqueRandom(int *array, int size) {
  int i;
  for (i = 0; i < size; ++i) {
    do {
       // кездейсоқ санды іздеу. Егер ол тізімде бұрыннан бар болса, басқасы ізделеді
       array[i] = rand() % (size * 4);
    } while (isValueInArray(array, i, array[i]) == 1);
    // бірегей мән табылған кезде, тізімдегі келесі позиция үшін санды іздейміз
  }
  // сол массивке сілтемені қайтарады
  return array;
}

// өлшемдері height * width болатын  matrix матрицасын экранға шығару
void print Matrix (int **matrix, int height, int width) {
  int i, j;
  for (i = 0; i < height; ++i) {
    for (j = 0; j < width; ++j)
      printf("%4d", matrix[i][j]);
    printf("\n");
```

```
  }
 }
 // матрицада value санының қанша рет орын алатынын санау
 int countValueInMatrix(int **matrix, int height, int width, int value) {
  int i;
  int result = 0;
  // әрбір жол - массив, әр жолдан қажетті элементті іздеп, оны жалпы санға қосамыз
  for (i = 1; i < height; ++i)
   result += isValueInArray(matrix[i], width, value);
  return result;
 }

 int main(int argc, char *argv[]) {
  // консольдегі орыс таңбаларын қолдау
  setlocale(LC_ALL, "Russian");
  // кездейсоқ сандар санауышын инициализациялау
  srand(time(0));
  // матрицаның өлшемдері
  int height = 10;
  int width = 10;
  int i;

  // матрица үшін жадтан орын бөлу
  int **matrix = (int**)malloc(sizeof(int*) * height);
  // матрица жолдарын кездейсоқ сандармен толтыру
  for (i = 0; i < height; ++i)
   matrix[i] = fillUniqueRandom((int*)malloc(sizeof(int) * width), width);

  // алынған матрицаны шығару
  printMatrix(matrix, height, width);

  // maxValue-ке ізделінді сан жазылады, maxValueCount-та - бұл санның неше рет кездесетіні  жазылады
  // алдымен, бұл - бірінші жолдың бірінші элементі деп есептейміз
  int maxValue = matrix[0][0];
  int maxValueCount = countValueInMatrix(matrix, height, width, maxValue);

  // бірінші жолдың барлық элементтерін қарастырамыз және әрқайсысының неше жиында кездесетінін санаймыз
```

```
  for (i = 1; i < width; ++i) {
    int thisValue = matrix[0][i];
    int thisValueCount = countValueInMatrix(matrix, height, width, thisValue);
    // егер көп жиында кездесетін сан табылса, оны maxValue айнымалысына
жазамыз
    if (maxValueCount < thisValueCount) {
      maxValueCount = thisValueCount;
      maxValue = thisValue;
    }
  }
  // нәтижені шығару
  printf("Ең көп кездесетін элемент %d. Ол %d рет кездеседі.\n",
    maxValue, maxValueCount);
    // жадыны босатамыз
  for (i = 0; i < height; ++i)
    free(matrix[i]);
  free(matrix);
  // соңы
  system("pause");
  return 0;
}
```



4 Жазықтықта нүктелер жиыны А және шеңберлер жиыны В берілген. А жиынынан мынадай әртүрлі екі нүктені табу керек: бұл екі нүкте арқылы өтетін түзу, В жиынындағы шеңберлердің максималды санымен қиылысатындай болуы керек.

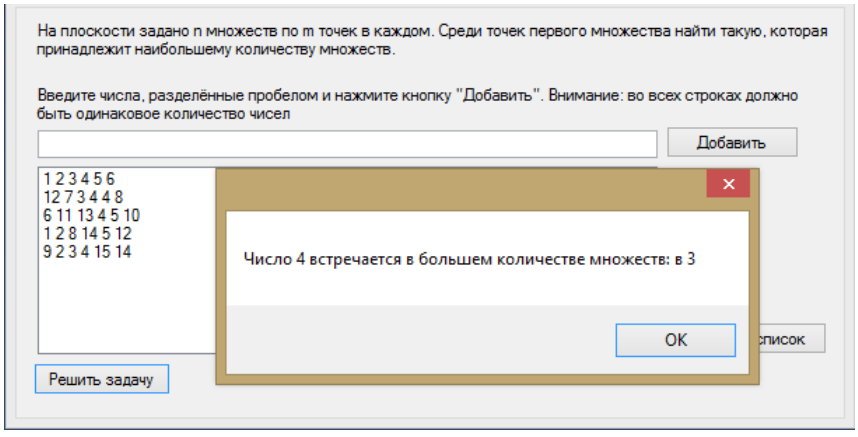



```
#include <iostream>
#include <cstdlib>
#include <clocale>
#include <ctime>
#include <cmath>
using namespace std;

// " point" құрылымы – бір нүктенің координаталар жұбын сақтайды
struct point {
    double x,y;
```

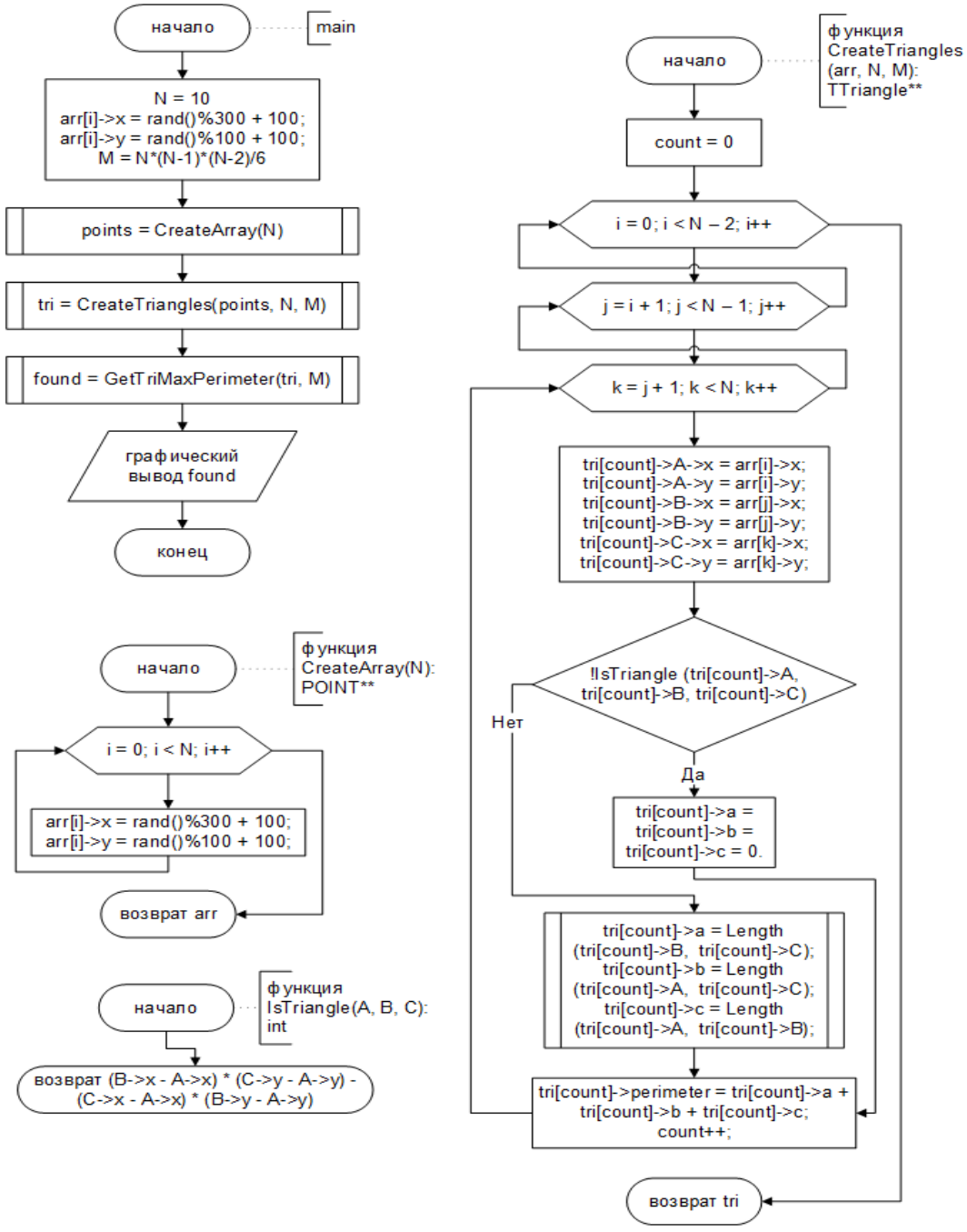
```
};

// "окружность"құрылымы  - центр координаты мен радиусын сақтайды
struct okr {
   double x,y,r;
};

// нүктелер мен шеңберлер саны
const int nmax=20;

// a және b нүктелері арқылы өтетін түзу c шеңберін қиып өтсе, "ақиқат" мәнін қайтарады. Функция соңында сипатталған
bool Peres(point a, point b, okr c);

int main() {
   // консольдегі кириллица қолдауы
   setlocale(0,"");

   // нүктелер мен шеңберлердің массивтерін, сондай-ақ басқа айнымалыларды жариялау
   point a[nmax];
   okr b[nmax];
   int n,m,i,j,k,p,mx,imx,jmx;

   // кездейсоқ сандар санауышын инициализациялау
   srand(time(0));

   // нүктелер санын енгізу
   do {
      cout << "Нуктелер саны дейін " << nmax << " n=";
      cin >> n;
   }while(n<1 || n>20);

   // шеңберлер санын енгізу
   do {
      cout << "Шенберлер саны дейін" << nmax << " m=";
      cin >> m;
   }while(m<1 || m>20);

   // N нүктелердің координаталарын кездейсоқ таңдау
```

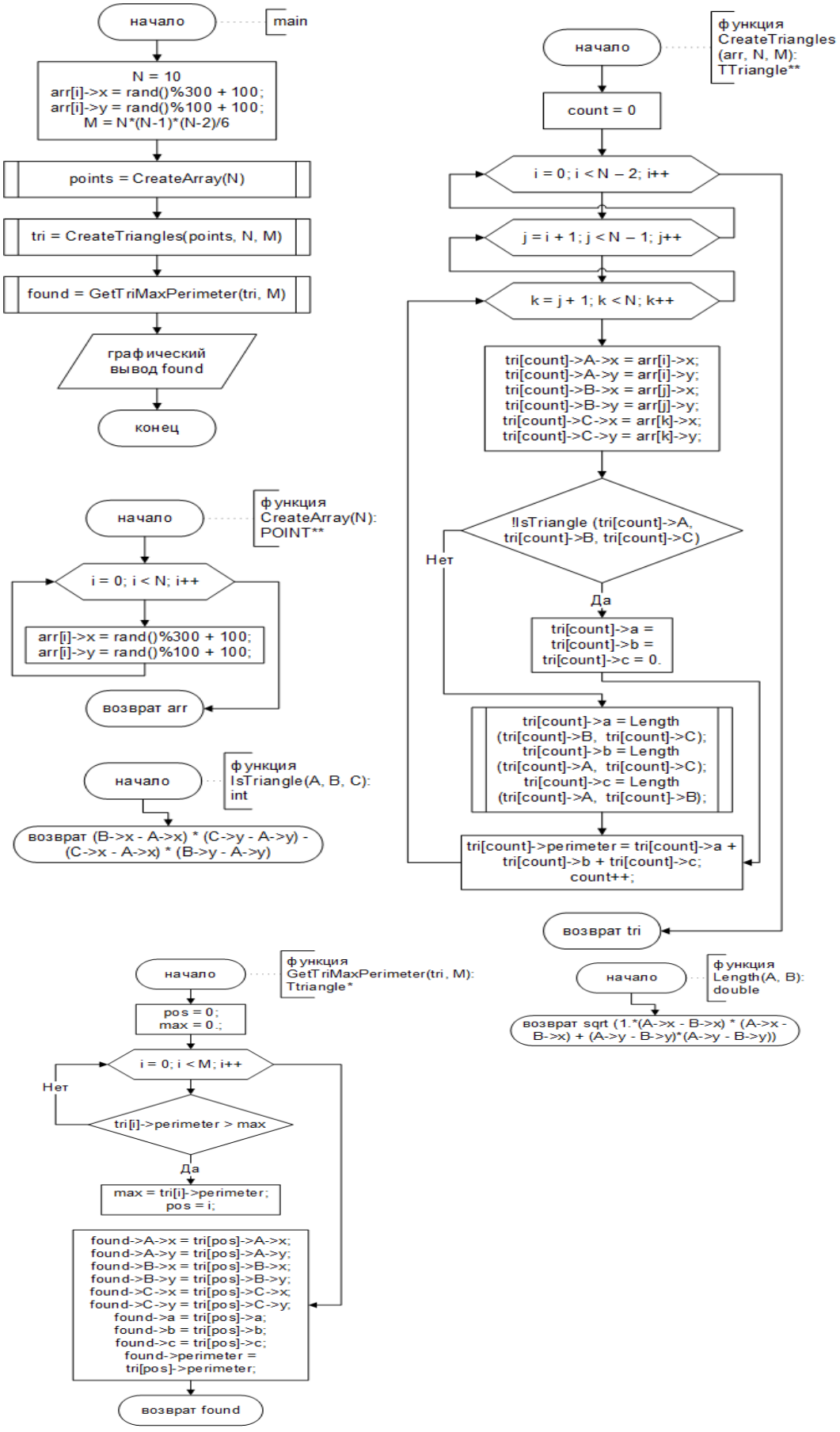
```
for(i=0; i<n; ++i) {
   a[i].x=rand()%30-10;
   a[i].y=rand()%30-10;
}

// M шеңберлерді кездейсоқ генерациялау
for(i=0; i<m; ++i) {
   b[i].x=rand()%30-10;
   b[i].y=rand()%30-10;
   b[i].r=5*rand();
}
// енгізілген сандарды шығару
cout << "Нуктелер координаталары:\nX:";
for(i=0;i<n;++i)
   cout << a[i].x << " ";
cout << "\nY:";
for(i=0;i<n;++i)
   cout << a[i].y << " ";
cout << "\n\nШеңберлер параметрлері:\nX:";
for(i=0;i<m;++i)
   cout << b[i].x << " ";
cout << "\nY:";
for(i=0;i<m;++i)
   cout << b[i].y << " ";
cout << "\nR:";
for(i=0;i<m;++i)
   cout << b[i].r << " ";
cout << "\n\n";
// негізгі бөлім
mx=0;
imx=0;
jmx=0;
// келесі екі цикл барлық нүктелер жұбын қарастырады
for(i=0;i<n-1;++i)
   for(j=i+1;j<n;++j) {
      k=0;
      // барлық шеңберлерді қарастырып,  бір түзумен қиылыстардың санын
санаймыз
      for(p=1;p<m;++p)
         if(Peres(a[i],a[j],b[p]))
```

```
                ++k;
            // mx-ке қиылыстардың көп санын, imx пен jmx-ке - қарастырылатын
түзу өтетін нүктелердің санын жазамыз
            if(k>mx) {
               mx=k;
               imx=i;
               jmx=j;
            }
          }
        // егер mx = 0, онда бірде бір түзу ешқандай шеңберді кесіп өтпейді
        if(mx==0)
          cout << "Киылысатын тузулер мен шенберлер жок\n";
        else
          // егер қиылыстар болса-нәтижені шығару
          cout << "Тузулер мен шенберлердің киылысуының максималды саны=" <<
mx
            << "\nБұл түзу келесі нүктелер аркылы өтеді
("<<a[imx].x<<';'<<a[imx].y<<") и ("<<a[jmx].x<<';'<<a[jmx].y<<")\n";
        cout << "\n";
        // соңы
        system("pause");}
      bool Peres(point a, point b, okr c) {
        double s, ab, h;
        // геометрия формулаларын қолданып, түзу мен шеңбердің қиылысуын
анықтаймыз
        s = abs(a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y));
        ab = sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
        h = s/ab;
        return (h<c.r); }
```



5 Нүктелер жиыны жазықтығы берілген. Бұл жиынның ішінен мынадай үш нүктені табу керек: осы үш нүкте арқылы, периметрі ең үлкен болатын үшбұрыш құралуы керек.




начало
main
N = 10
arr[i]->x = rand()%300 + 100;
arr[i]->y = rand()%100 + 100;
M = N*(N-1)*(N-2)/6
points = CreateArray(N)
tri = CreateTriangles(points, N, M)
found = GetTriMaxPerimeter(tri, M)
графический вывод found
конец
начало
функция CreateArray(N): POINT**
i = 0; i < N; i++
arr[i]->x = rand()%300 + 100;
arr[i]->y = rand()%100 + 100;
возврат arr
начало
функция IsTriangle(A, B, C): int
возврат (B->x - A->x) * (C->y - A->y) - (C->x - A->x) * (B->y - A->y)
начало
функция CreateTriangles (arr, N, M): TTriangle**
count = 0
i = 0; i < N – 2; i++
j = i + 1; j < N – 1; j++
k = j + 1; k < N; k++
tri[count]->A->x = arr[i]->x;
tri[count]->A->y = arr[i]->y;
tri[count]->B->x = arr[j]->x;
tri[count]->B->y = arr[j]->y;
tri[count]->C->x = arr[k]->x;
tri[count]->C->y = arr[k]->y;
!IsTriangle (tri[count]->A, tri[count]->B, tri[count]->C)
Нет
Да
tri[count]->a = tri[count]->b = tri[count]->c = 0.
tri[count]->a = Length (tri[count]->B, tri[count]->C);
tri[count]->b = Length (tri[count]->A, tri[count]->C);
tri[count]->c = Length (tri[count]->A, tri[count]->B);
tri[count]->perimeter = tri[count]->a + tri[count]->b + tri[count]->c;
count++;
возврат tri
начало
функция GetTriMaxPerimeter(tri, M): Ttriangle*
pos = 0;
max = 0.;
i = 0; i < M; i++
Нет
tri[i]->perimeter > max
Да
max = tri[i]->perimeter;
pos = i;
found->A->x = tri[pos]->A->x;
found->A->y = tri[pos]->A->y;
found->B->x = tri[pos]->B->x;
found->B->y = tri[pos]->B->y;
found->C->x = tri[pos]->C->x;
found->C->y = tri[pos]->C->y;
found->a = tri[pos]->a;
found->b = tri[pos]->b;
found->c = tri[pos]->c;
found->perimeter = tri[pos]->perimeter;
возврат found
начало
функция Length(A, B): double
возврат sqrt (1.*(A->x - B->x) * (A->x - B->x) + (A->y - B->y)*(A->y - B->y))

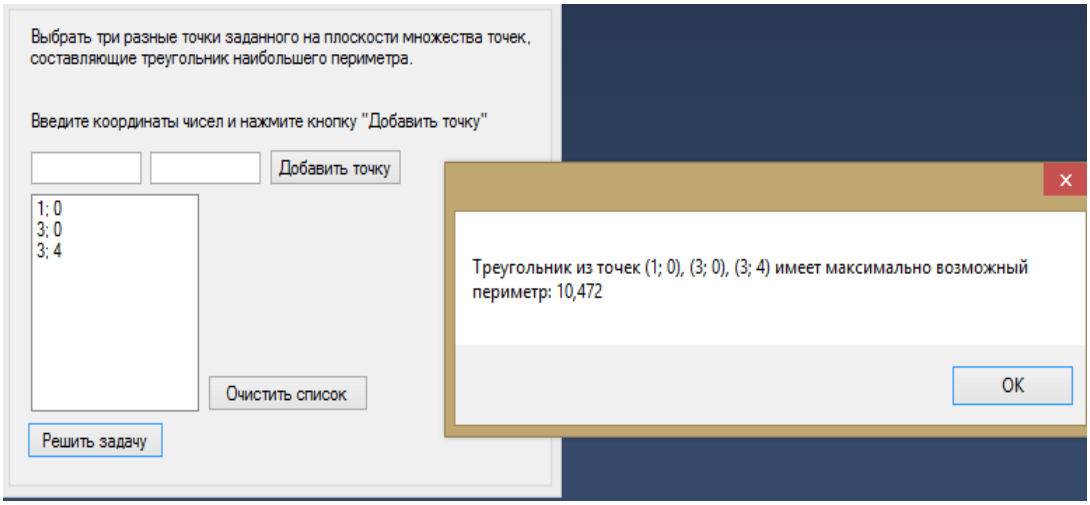
```
#include <iostream>
#include <ctime>
#include <cmath>
#include <Windows.h>    // сурет салуға арналған кітапхана

// жазықтықта N кездейсоқ нүктені жасайды, олардың массивіне сілтемені қайтарады
POINT **CreateArray (int N)
{   int i;
    POINT **arr = new POINT*[N];
    for (i = 0; i<N; i++)
    {   arr[i] = new POINT;
        arr[i]->x = rand()%300 + 100;
        arr[i]->y = rand()%100 + 100;
    }
    return arr;
}

// есептеулер аяқталғаннан кейін, нүктелік массив үшін бөлінген жадты босату
void DeleteArray (POINT** arr, int N)
{   for (int i = 0; i<N; i++)
        delete arr[i];
    delete [] arr;
}

// консольдегі нүктелер массивінің суретін салу
void ShowArray (HDC hDC, POINT** arr, int N, COLORREF color)
{   // текст "Координаты: "
    printf("Coordinaty:\n");
    // нүктені крест түрінде салу – яғни, нақты нүкте және төрт диагоналі
    for (int i = 0; i<N; i++)
    {   SetPixel(hDC, arr[i]->x, arr[i]->y, color);
        SetPixel(hDC, arr[i]->x-1, arr[i]->y-1, color);
        SetPixel(hDC, arr[i]->x+1, arr[i]->y+1, color);
        SetPixel(hDC, arr[i]->x-1, arr[i]->y+1, color);
        SetPixel(hDC, arr[i]->x+1, arr[i]->y-1, color);
        // мәтіндегі әрбір нүктенің координаталарын көрсету
        printf("(%d, %d)\n", arr[i]->x, arr[i]->y);
    }
```

```
}

// " TTriangle " құрлымы - үш нүкте, үш жақтың ұзындығы және периметрі
typedef struct _TTriangle
{   POINT *A, *B, *C;   // " POINT " құрылымы - төбелер
    double a,b,c;   // үшбұрыштың жақтарының ұзындағы
    double perimeter;
}   TTriangle;

// егер үш нүкте бір түзуде жатса, 0 қайтарады (яғни үш нүкте үшбұрыш жасамайды). Әйтпесе нөлдік емес санды қайтарады
int IsTriangle (POINT *A, POINT * B, POINT *C)
{   return (B->x - A->x) * (C->y - A->y) - (C->x - A->x) * (B->y - A->y);
}

// екі нүктенің арасындағы қашықтық
double Length (POINT *A, POINT * B)
{   return sqrt (1.*(A->x - B->x) * (A->x - B->x) + (A->y - B->y)*(A->y - B->y));
}

// N нүкте деректерінен барлық мүмкін үшбұрыштарды құру
TTriangle **CreateTriangles (POINT** arr, int N, int M)
{   // үшбұрыштардың қажетті санын жариялау, төбелер үшін бос орындар жасау
    int count = 0, i, j, k;
    TTriangle **tri = new TTriangle*[M];
    for (i = 0; i<M; i++)
    {   tri[i] = new TTriangle;
        tri[i]->A = new POINT;
        tri[i]->B = new POINT;
        tri[i]->C = new POINT;
    }

    // барлық нүктелердің үштіктерін қарастырамыз
    for (i = 0; i<N-2; i++)
        for (j = i+1; j<N-1; j++)
            for (k = j+1; k<N; k++)
            {   // нүктелердің координаттарын " TTriangle " объектісіне жазамыз
                tri[count]->A->x = arr[i]->x; tri[count]->A->y = arr[i]->y;
                tri[count]->B->x = arr[j]->x; tri[count]->B->y = arr[j]->y;
```

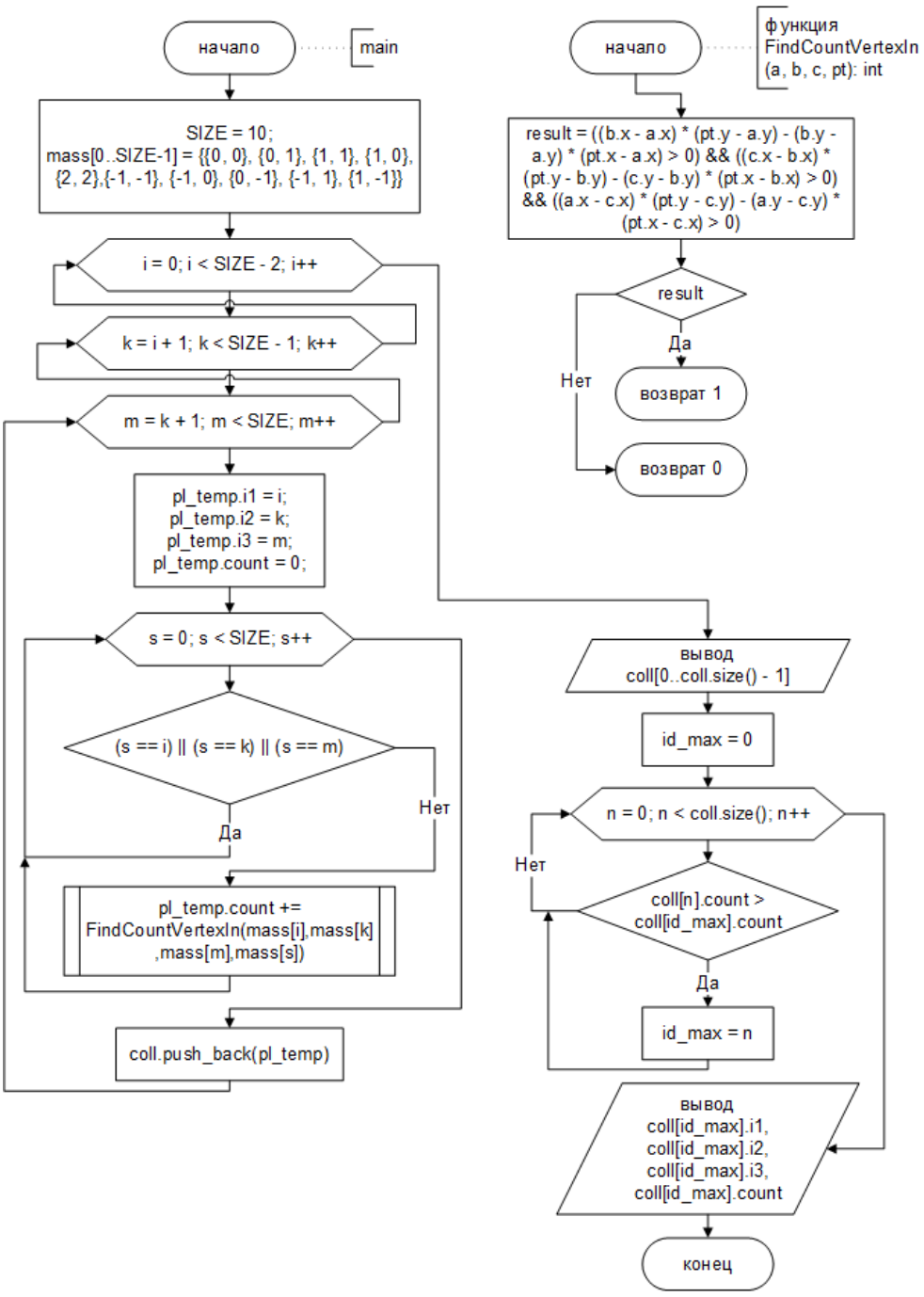
```
            tri[count]->C->x = arr[k]->x; tri[count]->C->y = arr[k]->y;

            // егер үш нүкте бір түзуде болса, жақтарын 0-ге келтіру
            if(!IsTriangle (tri[count]->A, tri[count]->B, tri[count]->C))
               tri[count]->a = tri[count]->b = tri[count]->c = 0.;
            else
            // әйтпесе үш жақтың ұзындығын есептейміз
            {   tri[count]->a = Length (tri[count]->B,  tri[count]->C);
               tri[count]->b = Length (tri[count]->A,  tri[count]->C);
               tri[count]->c = Length (tri[count]->A,  tri[count]->B);
            }
            // үшбұрыштың периметрін есептейміз
            tri[count]->perimeter = tri[count]->a + tri[count]->b + tri[count]->c;
            // келесі үшбұрышқа көшу
            count++;
         }

   // үшбұрыштар массивіне сілтемені қайтарамыз
   return tri;
}

// бір үшбұрышқа бөлінген жадты босату
void DeleteTri (TTriangle *t)
{   delete t->A;
   delete t->B;
   delete t->C;
   delete t;
}

// Үшбұрыштар массивіне бөлінген жадты босату
void DeleteTriangles (TTriangle **tri, int M)
{   for (int i = 0; i<M; i++)
      DeleteTri (tri[i]);
   delete [] tri;
}

// ең үлкен периметрі бар үшбұрышты табу
TTriangle *GetTriMaxPerimeter (TTriangle **tri, int M)
{   int pos = 0, i = 0;
   double max = 0.;
```

```
        // периметрдің ең үлкен мәнін іздейміз, үшбұрыштың нөмірін pos-қа
жазамыз
        for ( ; i<M; i++)
           if (tri[i]->perimeter > max)
           {   max = tri[i]->perimeter;
               pos = i;
           }
        // " TTriangle " объектісін жасап, табылған үшбұрыштың параметрлерін
жазамыз
        TTriangle *found = new TTriangle;
        found->A = new POINT;
        found->B = new POINT;
        found->C = new POINT;
        // нүктелердің координаттарын жазамыз
        found->A->x = tri[pos]->A->x; found->A->y = tri[pos]->A->y;
        found->B->x = tri[pos]->B->x; found->B->y = tri[pos]->B->y;
        found->C->x = tri[pos]->C->x; found->C->y = tri[pos]->C->y;
        // үшбұрыштың жақтарының ұзындығын жазамыз
        found->a = tri[pos]->a;
        found->b = tri[pos]->b;
        found->c = tri[pos]->c;
        // периметрді жазамыз
        found->perimeter = tri[pos]->perimeter;

        // үшбұрышты қайтарамыз
        return found;
    }

    // консольде Үшбұрыш салу
    void ShowTriangle (HDC hDC, TTriangle *t)
    {   // сурет салу үшін "қалам" инициализациясы
        HPEN hPen = CreatePen(PS_SOLID, 3, RGB(255,0,0));
        SelectObject(hDC, hPen);
        POINT p;

        // үшбұрыштың қабырғаларының үш жағы
        GetCurrentPositionEx(hDC, &p);
        MoveToEx(hDC, t->A->x, t->A->y, &p);
        LineTo (hDC, t->B->x, t->B->y);
        LineTo (hDC, t->C->x, t->C->y);
```

```
        LineTo (hDC, t->A->x, t->A->y);

        // үшбұрыштың параметрлерін жазу: периметрі мен координаталары
        printf("\n\n\n\n\n\nMax perim: %.3lf\n", t->perimeter);
        printf("ushburish:\n(%d, %d)\n(%d, %d)\n(%d, %d)\n", t->A->x, t->A->y, t->B->x, t->B->y, t->C->x, t->C->y);     // қаламды босатамыз
        DeleteObject (hPen);
    }
    int main()
    {   // нүктелердің саны
        int N = 10; // сурет салу консолін инициализациялау
        HDC hDC = GetDC(GetConsoleWindow());
        srand ((unsigned int) time (NULL));// нүктелерді құру және шығару
        POINT **points = CreateArray (N);
        ShowArray (hDC, points, N, RGB (255,255,0));  // M - M нүктелерінен барлық үштіктердің саны
        int M = N*(N-1)*(N-2) / 6;// үшбұрыштар жасаймыз
        TTriangle **tri = CreateTriangles (points, N, M);  // ізделінді үшбұрышты тауып, сызамыз
        TTriangle *found = GetTriMaxPerimeter (tri, M);
        ShowTriangle (hDC, found);// қажет емес жадты босату
        DeleteArray (points, N);
        DeleteTriangles (tri, M);
        DeleteTri (found);// Enter пернесін басқан кезде консольді жабу
        getchar();  // соңы
        return 0;
    }
```



6 Жазықтықта нүктелер жиыны берілген. Бұл жиыннан мына шартты қанағаттандыратын үш түрлі A, B, C нүктелерін табу керек: ABC үшбұрышының ішінде осы жиынның нүктелерінің максималды саны орналасуы керек.



#include <iostream>

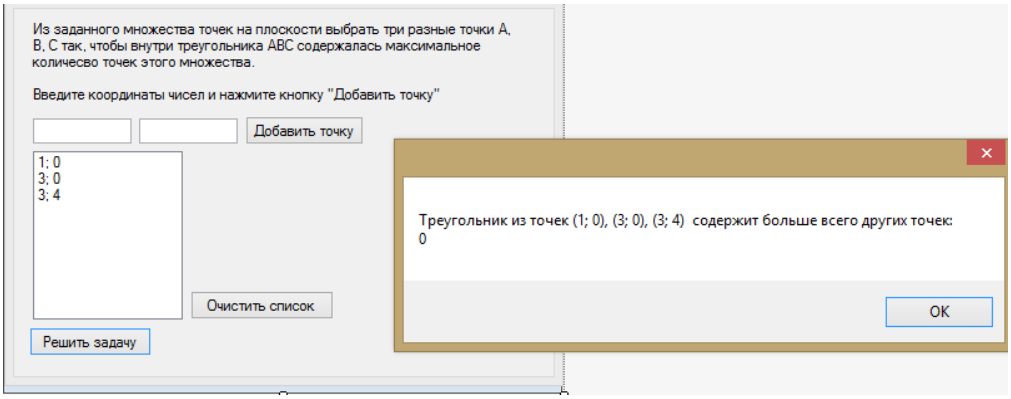
#include <vector> // векторлармен жұмыс істеуге арналған кітапхана (массивтердің аналогы)

#include <conio.h>

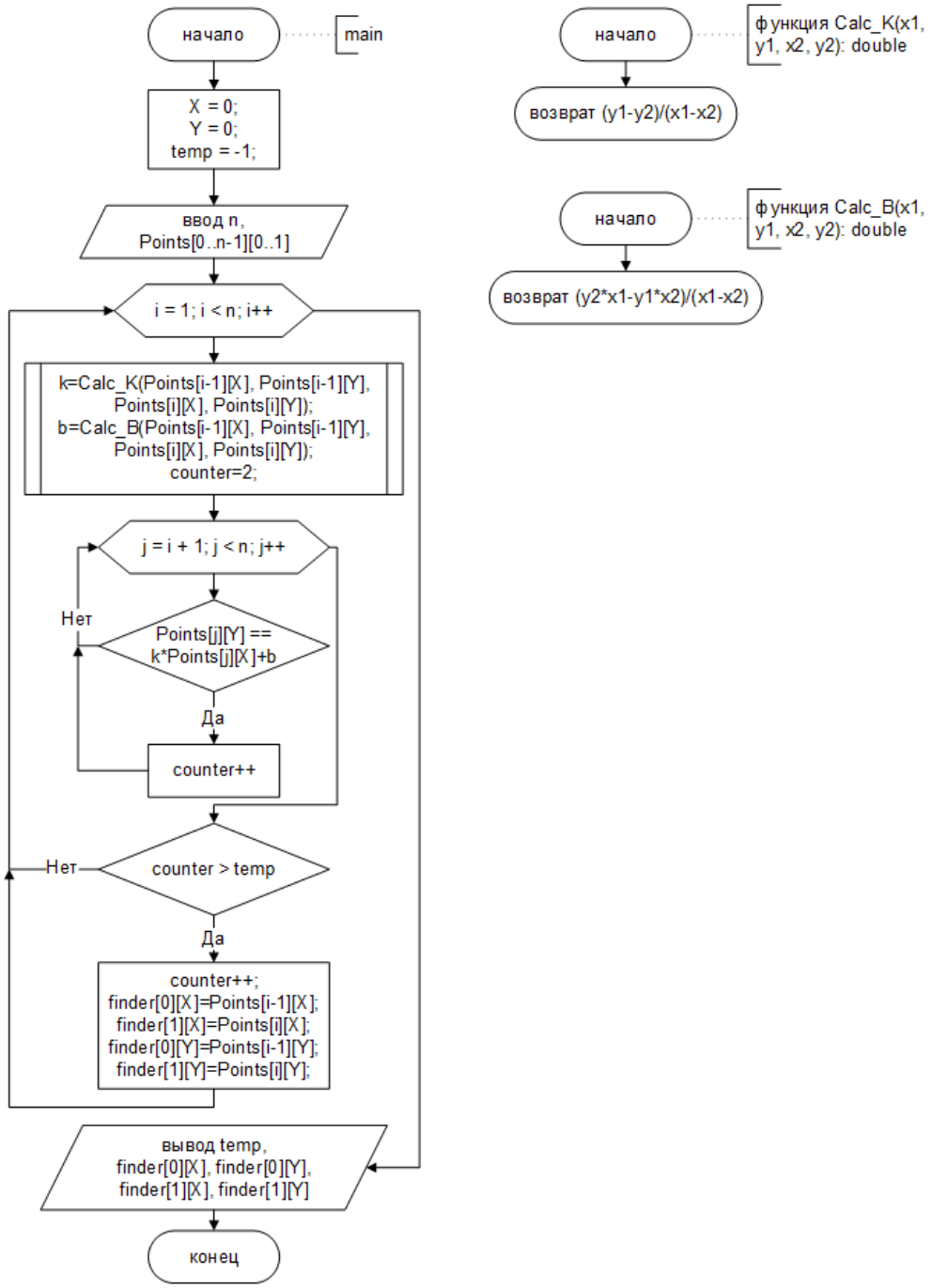
```
// " point " құрылымы – координаталар жұбы
struct point {
   double x;
   double y;
}; // "Многоугольник" құрылымы - үш төбенің нөмірі және үшбұрыштың ішіндегі нүктелер саны
struct polygon {
   int i1;
   int i2;
   int i3;
   int count;
}; // егер төртінші нүкте үш нүктеден құралған үшбұрышқа жатса, 1 - ді қайтарады, әйтпесе-0
int FindCountVertexIn(const point& a, const point& b, const point& c, const point& pt) {
   // тиістілікті тексеру үшін геометриялық формула барлық төрт нүктенің екі координатын да пайдаланады
   if (
      ((b.x - a.x) * (pt.y - a.y) - (b.y - a.y) * (pt.x - a.x) > 0) &&
      ((c.x - b.x) * (pt.y - b.y) - (c.y - b.y) * (pt.x - b.x) > 0) &&
      ((a.x - c.x) * (pt.y - c.y) - (a.y - c.y) * (pt.x - c.x) > 0)
   )
      return 1;
   return 0;
}
int main() {  // консольде кириллицаны қолдау
   setlocale(LC_ALL, "Russian");  // нүктелер массивінің өлшемі
   const int SIZE = 10;   // нүкте координаттары
   point mass[SIZE] = {
                  {0, 0}, {0, 1}, {1, 1}, {1, 0}, {2, 2},
                  {-1, -1}, {-1, 0}, {0, -1}, {-1, 1}, {1, -1}
                  };
   // “Көпбұрыш” түріндегі аралық айнымалы (шын мәнінде үшбұрыш)
   polygon pl_temp;  // үшбұрыштардың векторы (яғни массив).
   std::vector<polygon> coll;   // нүктелердің барлық мүмкін үштіктерін, яғни үшбұрыштарды санау
   for(int i = 0; i < SIZE - 2; ++i)
      for(int k = i + 1; k < SIZE - 1; ++k)
         for(int m = k + 1; m < SIZE; ++m) {    // нүкте нөмірлерін жазу
pl_temp.i1 = i;
```

```
                pl_temp.i2 = k;
                pl_temp.i3 = m;
                pl_temp.count = 0; // үшбұрышта басқа нүктелер бар-жоғын тексеру
                for(int s = 0; s < SIZE; ++s) {
                   if (
                      (s == i) ||
                      (s == k) ||
                      (s == m)
                      )  // төртінші нүкте үшбұрыштың төбесімен сәйкес келсе, тексермейміз, циклден шығамыз
                         continue;  // әйтпесе, біз тиесілілікті тексеріп, тиесілі болған кезде жалпы санға 1 қосамыз
                      pl_temp.count += FindCountVertexIn(
                                          mass[i],
                                          mass[k],
                                          mass[m],
                                          mass[s]
                                          );
                } // push_back әдісі pl_temp көшірмесін массивке жібереді (вектор coll). Көшірмені жасаймыз, себебі pl_temp айнымалысын әрі қарай өзгертеміз
                   coll.push_back(pl_temp);
                }
          // барлық үшбұрыштарды және олардың ішіндегі нүктелер санын көрсету
          std::cout << "Барлық мумкін нұсқалар:\n";
          for(int n = 0; n < coll.size(); ++n)
             std::cout << coll[n].i1 << ' ' << coll[n].i2 << ' ' << coll[n].i3 << ' ' << coll[n].count << '\n';
          std::cout <<
"=========================================\n";
          // нүктелердің ең көп санына ие болатын үшбұрыштың индексін coll-де табамыз (id_max)
          int id_max = 0;
          for(int n = 0; n < coll.size(); ++n)
             if (coll[n].count > coll[id_max].count)
                id_max = n;  // нәтижені шығару
          std::cout << "ushburysh(x, y, z, count) : ";
          std::cout << coll[id_max].i1 << ' '
                << coll[id_max].i2 << ' '
                << coll[id_max].i3 << ' '
```

```
        << coll[id_max].count << '\n';   // Enter пернесін басқан кезде консольді
жабу
    system("pause");  // соңы
    return 0;
}
```



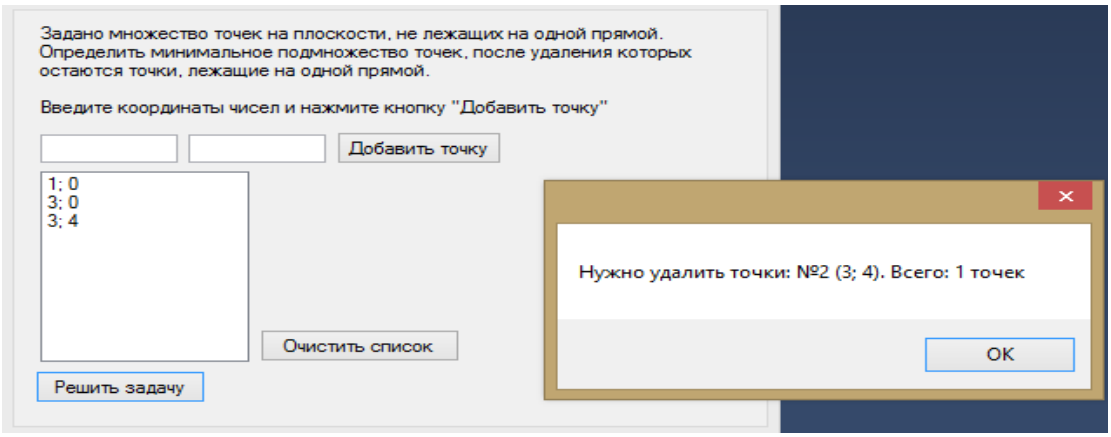
7 Жазықтықтағы бір түзудің бойында жатпайтын нүктелер жиыны берілген. Келесі шартты қанағаттандыратын, нүктелердің ең аз жиынын анықтаңыз: осы анықталған жиынды алып тастағаннан кейін бір түзуде жатқан нүктелер ғана қалады.



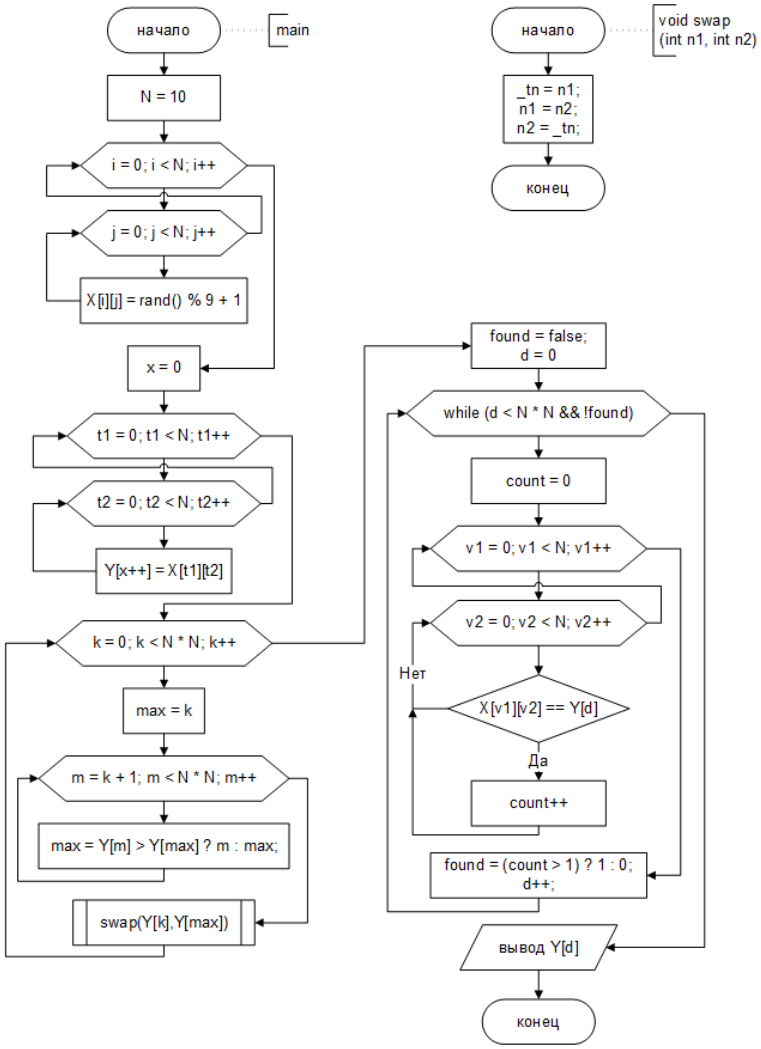

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
 // препроцессорлық директивалар - барлық X-ті 0-ге және Y-ді 1-ге ауыстыру
#define X 0
```

```
#define Y 1
 // (x1, y1) и (x2, y2)  нүктелері арқылы өтетін y = Kx + B түзуінің теңдеуі үшін K коэффициентін табады.
double Calc_K(double x1, double y1, double x2, double y2)
{
    return (y1-y2)/(x1-x2);
}
 // аналогты коэффициент B
double Calc_B(double x1, double y1, double x2, double y2)
{
    return (y2*x1-y1*x2)/(x1-x2);
}
 int main ()
{// айнымалыларды жариялау
    double k,b, finder[2][2];
    int counter, temp=-1, n;  // нүктелер санын енгізу (үштен кем емес)
    printf ("Координаталар санын енгізіңіз ( n>2 ): ");
    scanf ("%d", &n);
    if(n<=2) { printf("Координаталар саны оте аз"); _getch(); _exit (EXIT_FAILURE);}
     // N * 2 матрицасы үшін (әрқайсысының екі координаты бар N нүктелер) жадтан орын бөлу  және олардың кординаттарын шығарамыз
    double ** Points = (double**) malloc(n * sizeof (double*));
    for (int i=0; i<n; i++)
    {
        Points[i]= (double*)malloc(2 * sizeof(double));
        printf("Жұпты енгізіңіз[%d]:\nx: ", i + 1); // X нүктесінің координаттарын енгізу
        scanf("%f", &Points[i][X]);
        printf("y: ");   // Y нүктесінің координаттарын енгізу
        scanf("%f", &Points[i][Y]) ;
    }       // көрші нүктелердің барлық жұптарын қарастыру
    for (int i=1; i<n; i++)
    {  // жұп нүктелер үшін олар арқылы өтетін y = kx + B түзуінің теңдеуін табамыз
        k=Calc_K(Points[i-1][X], Points[i-1][Y], Points[i][X], Points[i][Y]);
        b=Calc_B(Points[i-1][X], Points[i-1][Y], Points[i][X], Points[i][Y]);
        // екі нүкте қазірдің өзінде осы түзуде жатыр деп есептейміз
        counter=2;    // осы түзуде жатқан қалған нүктелердің санын есептейміз
```

```
                for (int j=i+1; j<n; j++)
                {
                    if(Points[j][Y]==k*Points[j][X]+b) ++counter;
                }
                // temp айнымалысында көп санды, алғашқы екеуінің координаттарын
- finder матрицасында есте сақтаймыз
                if(counter>temp)
                {
                    temp=counter;
                    finder[0][X]=Points[i-1][X]; finder[1][X]=Points[i][X];
                    finder[0][Y]=Points[i-1][Y]; finder[1][Y]=Points[i][Y];
                }
            }
        // табылған мәндердің нәтижесі - бір түзудегі нүктелердің ең көп саны,
берілген түзудің алғашқы екі нүктесінің координаттары
            printf("координаталар: %d\n line:: x1: %f y1: %f\nx2: %f y2: %f", temp,
finder[0][X], finder[0][Y], finder[1][X], finder[1][Y]); // Enter пернесін басқан кезде
консольді жабу
            _getch();  // соңы
            return 0;}
```



8 Берілген матрицада бірнеше рет кездесетін сандардың максимумын табыңыз.



#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <memory.h> // жадты басқару кітапханасы

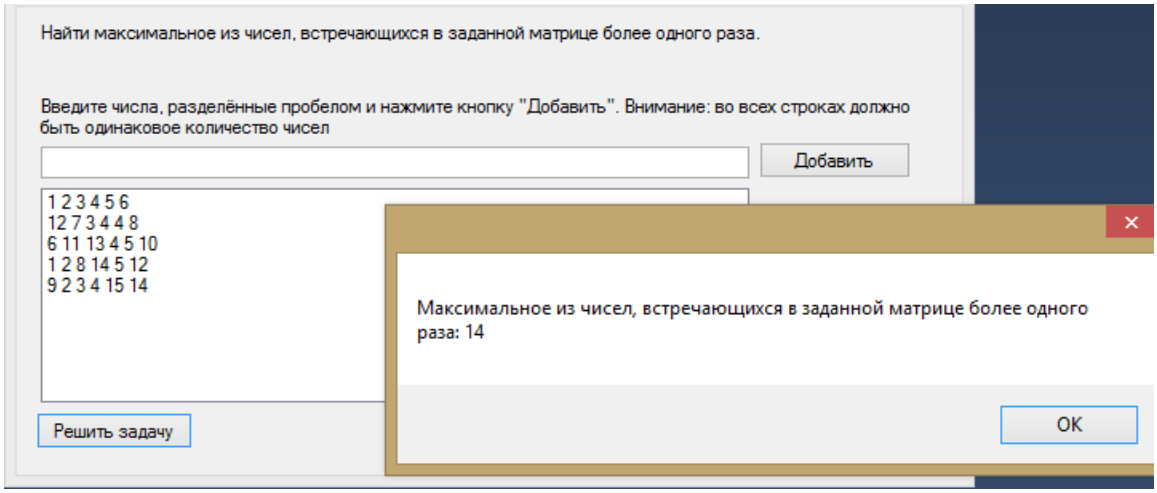
#define N 10 // квадрат матрицаның өлшемі

// мәндерді айырбастау процедурасы - сұрыптау кезінде қолданылады, соңында жүзеге асырылады
void swap(int& n1, int& n2);

int main()
{

```
// матрица - бұл жолдар массиві (олардың әрқайсысы массив болып табылады)
// жолдар массивін жариялаймыз
int** X = new int*[N];
// ол үшін жадтан орын бөлеміз
memset((void*)X, 0x00, 4 * N);

// жолдарды жариялаймыз және оларды кездейсоқ сандармен толтырамыз
for (int i = 0; i < N; i++)
{
   X[i] = new int[N];
   for (int j = 0; j < N; j++)
   {
      // rand() % 9 - 0-ден 8-ге дейінгі кездейсоқ сан
      // rand() % 9 - 1-ден 9-ға дейінгі кездейсоқ сан
      X[i][j] = rand() % 9 + 1;
      // жолдың элементтерін бос орын арқылы бірден жазу
      printf("%d ",X[i][j]);
   }
   // жолдың соңы - жаңа жолға өту
   printf("\n");
} // N * N элементтерінен Y массивін жариялаймыз, оған  X матрицасының элементтерін жазамыз
int x = 0, *Y = new int[N * N];
for (int t1 = 0; t1 < N; t1++)
   for (int t2 = 0; t2 < N; t2++)
      Y[x++] = X[t1][t2]; // Y массивін кему бойынша сұрыптаймыз
for (int k = 0; k < N * N; k++)
{
   int max = k;
   // егер k-ші элементтен кейін, одан үлкенірек сан тапсақ, оны k элементімен ауыстырамыз
   for (int m = k+1; m < N * N; m++)
      max = Y[m] > Y[max] ? m : max;
   swap(Y[k],Y[max]);
} // Y массиві кему реті бойынша сұрыпталғандықтан, X матрицасында бірнеше рет пайда болатын Y элементінің бірінші элементі максимум болады.
bool found = false; int d = 0;
while (d < N * N && !found) // found 1-ге тең болғанда, цикл үзіледі
{
```

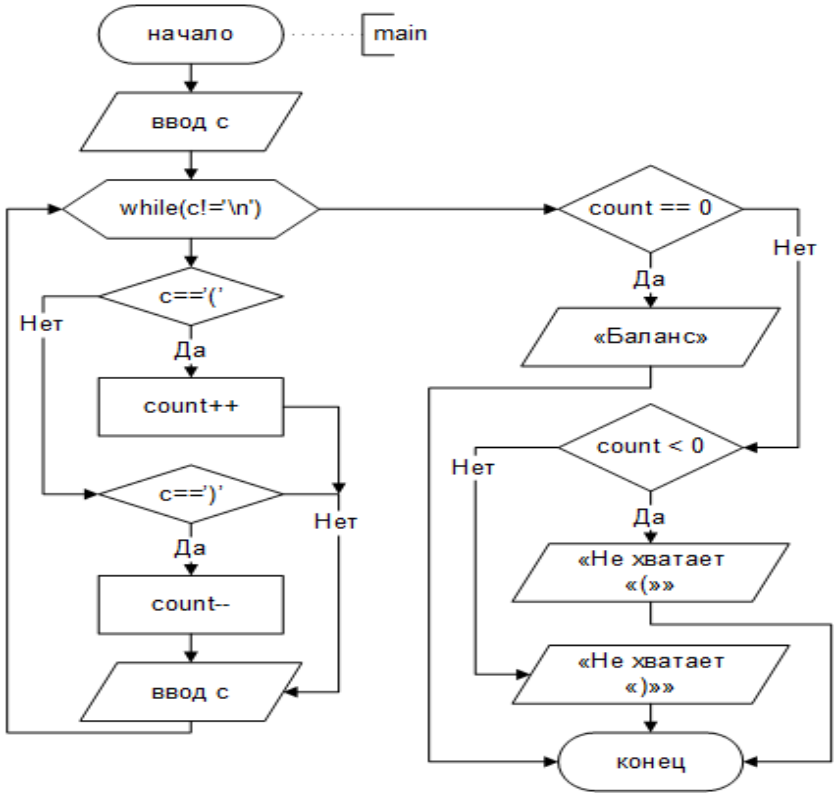
```
        int count = 0;
        // енгізілген санын (count)санаймыз
        for (int v1 = 0; v1 < N; v1++)
            for (int v2 = 0; v2 < N; v2++)
                if (X[v1][v2] == Y[d]) count++;
        // егер элемент бірнеше рет кірсе -found = 1- циклден шығу
        found = (count > 1) ? 1 : 0; d++;
    } // нәтижені шығару
    printf("\nMax = %d\n",Y[d]); // Enter пернесін басқан кезде консольден шығу
    _getch(); // соңы
    return 0;}
void swap(int& n1, int& n2)
{
    int _tn = n1;
    n1 = n2;
    n2 = _tn;
}
```



9 Берілген мәтінде ашылатын және жабылатын жақшалардың балансы бар-жоғын, яғни ашылатын және жабылатын жақшалардың келесі қасиеттермен бір-біріне сәйкестігін орнатуға болатындығы рас па:

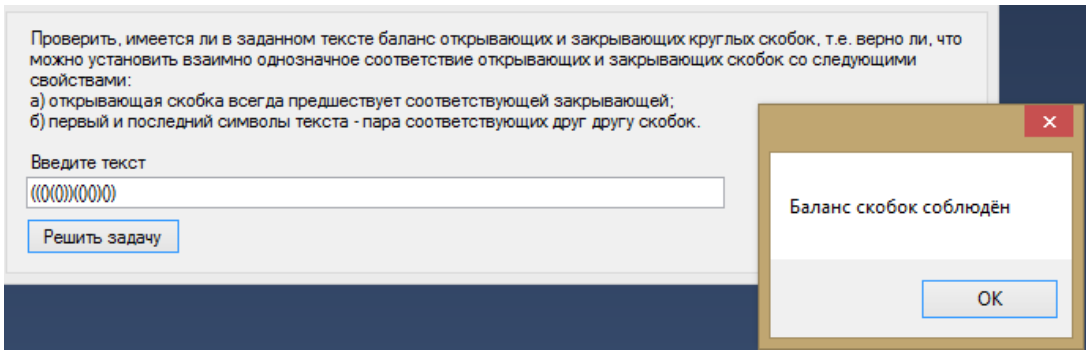
а) ашылатын жақша әрқашан тиісті жабылатын жақшаның алдында болады;

ә) мәтіннің бірінші және соңғы таңбалары бір-біріне сәйкес келетін жақшалар жұбы.

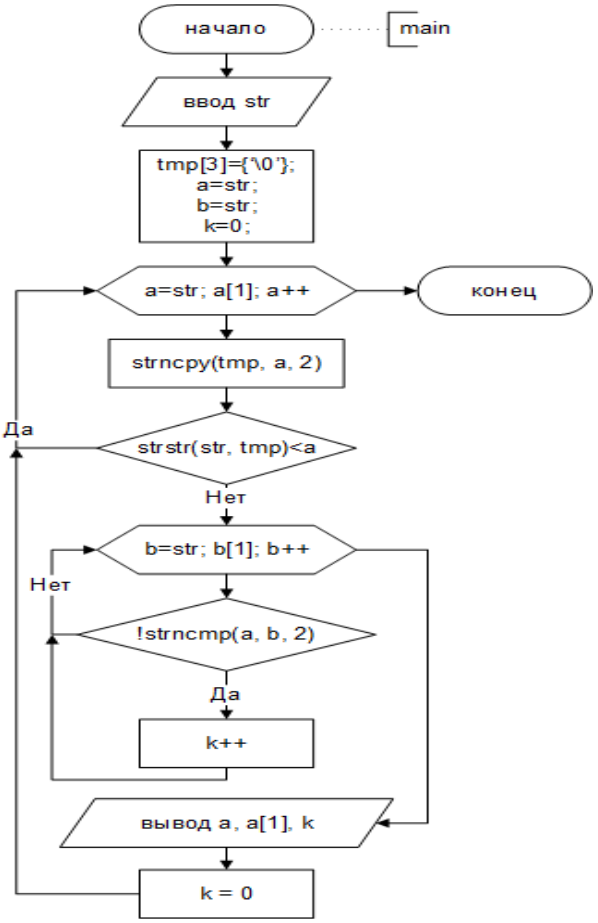



```
#include <stdio.h>
#include <conio.h>
#include <clocale>

int main()
{
   // консольдегі кириллица қолдауы
   setlocale(LC_ALL, "Russian");
   char c;
   int count = 0;

   // бастапқы жолды енгізу
   printf("Текстті енгізіңіз: \n");
   scanf("%c",&c);
```

```
// жолдың әрбір симолын жолдың соңына жеткенше тексереміз
while(c!='\n')
{
   // ашылатын жақша - count санын арттырамыз
   if(c=='(') count++;
   // жабу жақшасы - count азайту
   else if(c==')') count--;
   // жолдың келесі таңбасын оқу
   scanf("%c",&c);
}
if(count==0) printf("\nБаланс");    // count = 0 - жақшалар дұрыс
орналастырылған
else if(count<0) printf("\n( жетпейді");    // жабылатын жақшалары көп
else if(count>0) printf("\n) жетпейди");    // ашылатын жақшалары көп

// Enter пернесін басқан кезде консольді жабу
getch(); // соңы
return 0;}
```

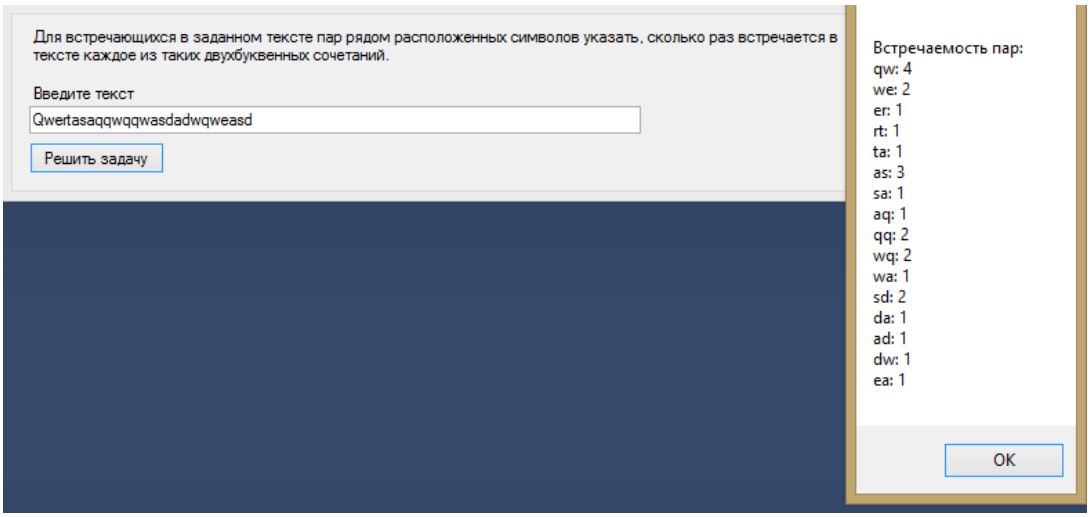


10 Берілген мәтінде кездесетін жұптар үшін, екі әріптен тұратын комбинациялардың әрқайсысы мәтінде қанша рет кездесетінін көрсетіңіз.

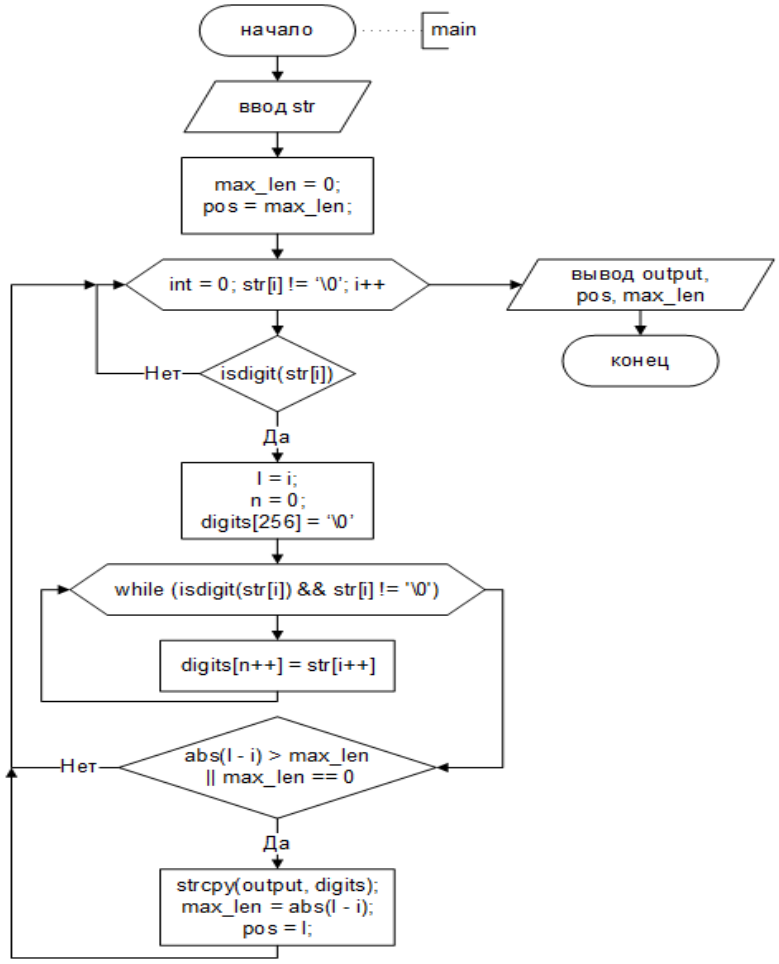



```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <clocale>

int main(){  // консольдегі кириллица қолдауы
        setlocale(LC_ALL, "Russian");
   char str[100], tmp[3]={'\0'}, *a=str, *b=str   // жолды енгізу
   puts("Сөзді енгізіңіз: ");
   gets(str); unsigned k=0;    // жолдың әрбір символын қарастыру
   for(a=str; a[1]; ++a){
      // tmp-ге көршілес таңбалар жұбын жазамыз
      strncpy(tmp, a, 2);
      // егер бұл жұп бұрын кездескен болса, онда бұдан кейінгі операторларды өткізіп жіберіп, келесі жұпқа өтеміз
      if(strstr(str, tmp)<a) continue;
      // жұптың бүкіл жолда қанша рет кездесетінін есептейміз
```

```
            for(b=str; b[1]; ++b){
               // strncmp(a, b, 2) егер бұл жұпты тапсақ, нөлге тең, сондықтан !strncmp(a, b, 2) = !0 = "ақиқат" және k есептегіші артады
               if(!strncmp(a, b, 2)) ++k;
            }
            // жұптың кездесу санын шығару
            printf("%c%c %u рет кездеседі\n", *a, a[1], k);
            // есептеуішті тазалаймыз
            k=0;
         }
         // Enter пернесін басқан кезде консольді жабу
         getch();
         // соңы
         return 0;
      }
```

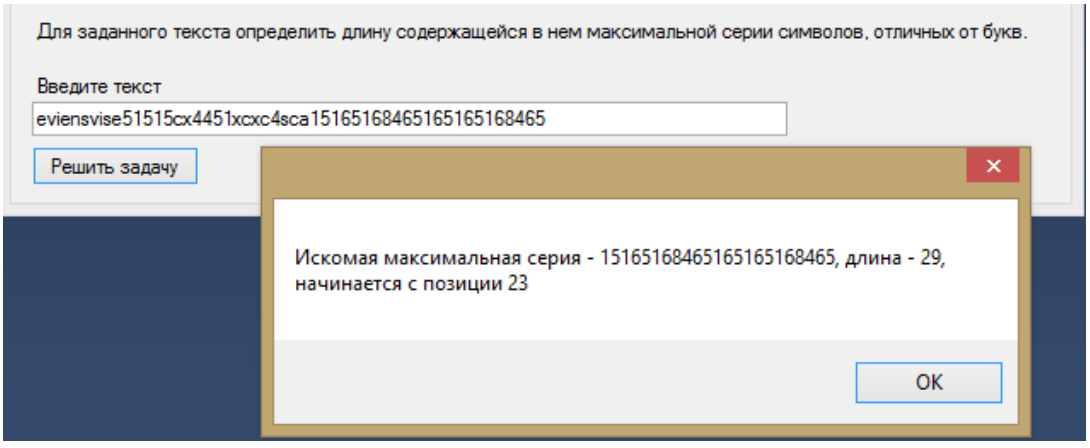


11 Берілген мәтін үшін, ондағы әріптерден басқа таңбалар сериясының максималды ұзындығын анықтаңыз.

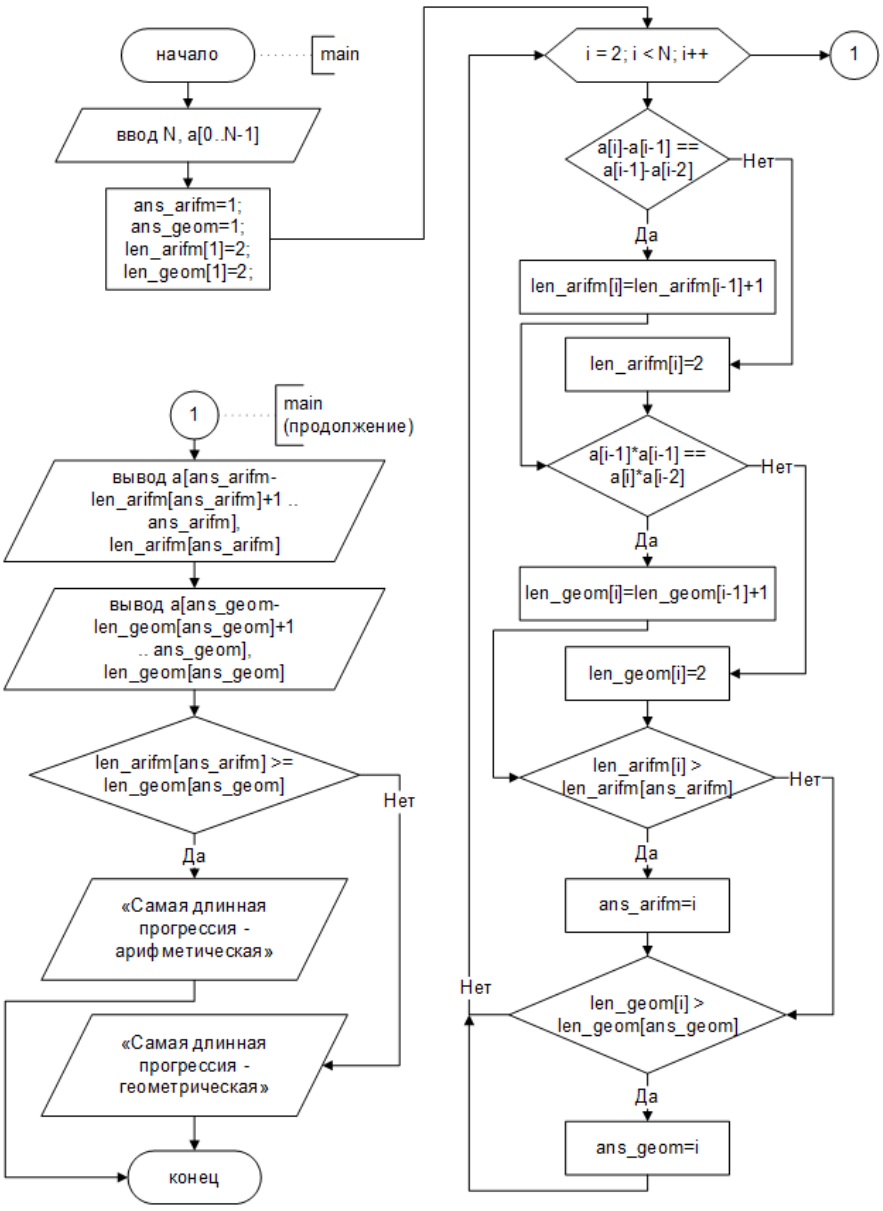



```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>  // символдармен жұмыс істеуге арналған кітапхана
 int main(int argc, char* argv[])
{
   // берілген жол
   static char str[256] = "abulkhair51515cx4451xcxc4sca1516516846516516516846 5\0";
    // жолды шығару
   printf("string = %s\n", str);
    // output жолын жариялаймыз ("\0 " - бастапқыда бос жол, 256 - таңбалар үшін осынша орын сақтаймыз), содан кейін оған нәтижені жазамыз
   char output[256] = "\0";
```

```
        int max_len = 0, pos = max_len;
        // перебор строки. і циклдің ішінде артатындықтан, әріптер өткізіп жіберіледі
        for (int i = 0; str[i] != '\0'; i++)
          // егер цифрді кездестірсеңіз
          if (isdigit(str[i]))
          {
            // і индексі l айнымалысында сақталады
            int l = i, n = 0; char digits[256] = "\0";
            // барлық келесі сандарды digit жолына ретімен жазамыз (әріп кездескенше немесе соңына жеткенше)
            while (isdigit(str[i]) && str[i] != '\0')
              digits[n++] = str[i++];
             // abs(l - i) – digits жолдың ұзындығы, цифрлар саны көп жолды есте сақтаймыз, оны output-ке жазамыз
            if (abs(l - i) > max_len || max_len == 0)
            {
              strcpy(output, digits);
              max_len = abs(l - i);
              pos = l;
            }
          }
        // ізделінді жолды (output), str жолын қандай символдан басталатынын (pos) және оның ұзындығын (max) шығарамыз.
        printf("output = %s pos = %d len = %d\n", output, pos, max_len);
         // Enter пернесін басқан кезде консольді жабу
        _getch();
        // соңы
        return 0;
    }
```



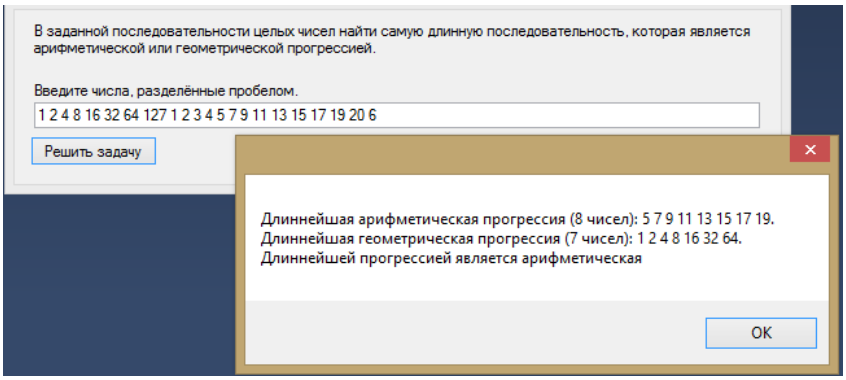
12 Берілген бүтін сандар тізбегінде арифметикалық немесе геометриялық прогрессия болып табылатын ең ұзын тізбекті табыңыз.



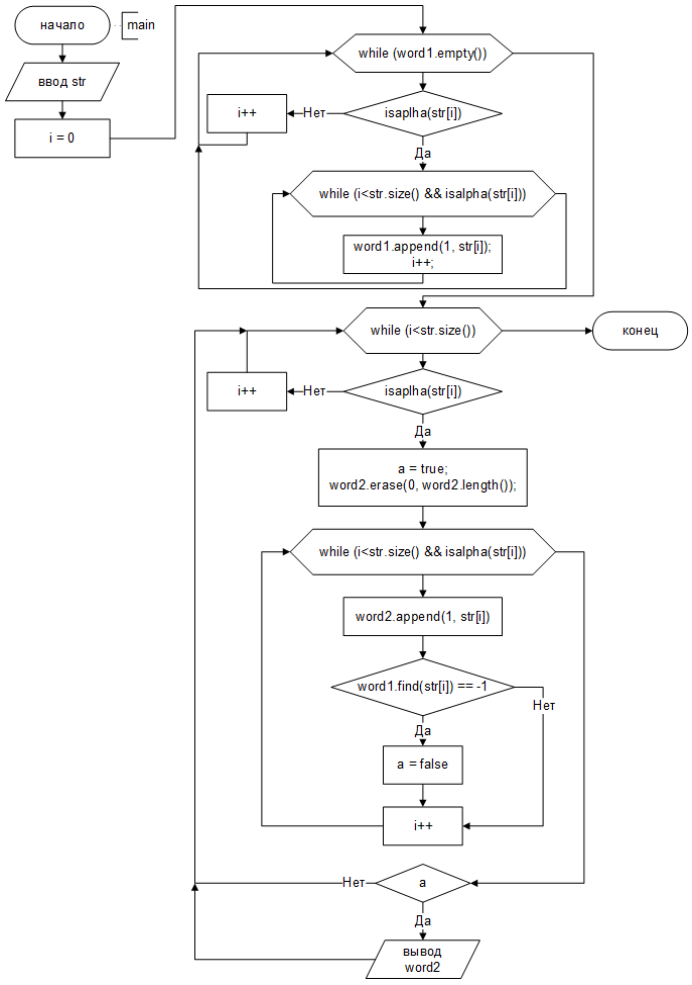

```
#include <iostream>
using namespace std;
 int main()
{
   // берілген тізбек
   int a[]={1,2,4,8,16,32,64,127,1,2,3,4,5,7,9,11,13,15,17,19,20,6};
   // көмекші айнымалылар
   // ans_arifm и ans_geom максималды сәйкес прогрессияның ұзындықтарын сақтайды
   int N=21, len_arifm[100],len_geom[100],ans_arifm=1,ans_geom=1,i;
```

```
// прогрессияның ұзындықтарын сақтауға арналған массивтер
len_arifm[1]=2;
len_geom[1]=2;
// берілген массив бойымен өту
for (i=2; i<N; ++i)
{
   // элемент арифметикалық прогрессияның жалғасы бола ма
   if(a[i]-a[i-1]==a[i-1]-a[i-2])
      // егер иә болса, санауышты арттырамыз
      len_arifm[i]=len_arifm[i-1]+1;
   // егер жоқ болса, санауышты 2-ге кемітеміз (өйткені кез-келген жұп сандарды прогрессия деп санауға болады)
   else len_arifm[i]=2;
   // геометриялық прогрессияны тексеру
   if(a[i-1]*a[i-1]==a[i]*a[i-2])
      len_geom[i]=len_geom[i-1]+1;
   else len_geom[i]=2;
   // арифметикалық прогрессияның үлкен ұзындығын есте сақтау
   if(len_arifm[i]>len_arifm[ans_arifm])
      ans_arifm=i;
   // жәнегеометриялық
   if(len_geom[i]>len_geom[ans_geom])
      ans_geom=i;
}
// нәтижелерді шығару
cout<<"En ulken arifmetikalyk progressiya: ";
// арифметикалық прогрессия
for (i=ans_arifm-len_arifm[ans_arifm]+1; i<=ans_arifm;++i)
      cout<<a[i]<<' ';
// оның ұзындығы
cout<<"Uzyndygy: "<<len_arifm[ans_arifm]<<endl;
// геометриялық прогрессия
cout<<"En ulken geometryalyk  progressiya: ";
for (i=ans_geom-len_geom[ans_geom]+1; i<=ans_geom;++i)
      cout<<a[i]<<' ';
// оның ұзындығы
cout<<"Uzyndygy: "<<len_geom[ans_geom]<<endl;
// прогрессиялардың қайсысы ұзынырақ
if (len_arifm[ans_arifm]>=len_geom[ans_geom])
     cout<<"Jauaby arifmetikalyk progressiya"<<endl;
```

```
    else cout<<"Jauaby geometriyalik progressiya"<<endl;
    // Enter пернесін басқан кезде консольді жабу
    system("PAUSE");
    // соңы
    return 0;
}
```

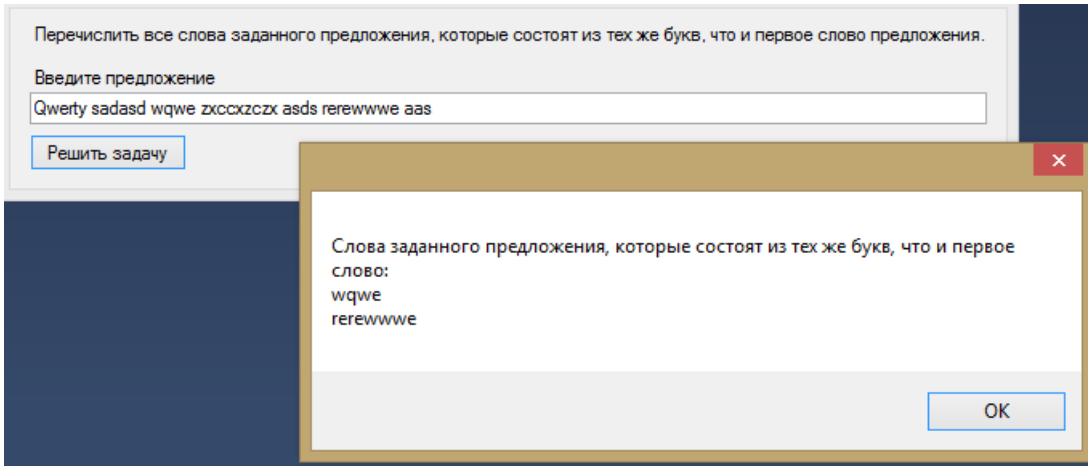


13 Берілген сөйлемнің бірінші сөзімен бірдей әріптерден тұратын барлық сөздерінің тізімін жасаңыз.

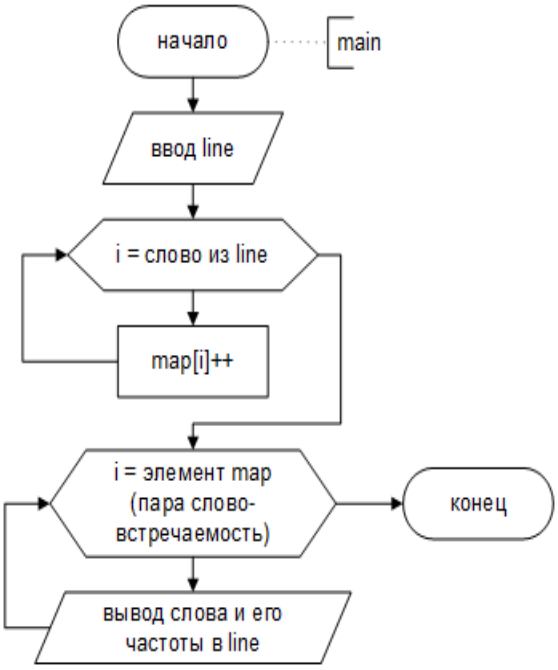


```
#include <iostream>
#include <string>
#include <cctype>
#include <clocale>  // әр түрлі кодировкадағы символдарды қолдау кітапханасы

using namespace std;

 int main()
{
   // консольдің кириллица символдарын қолдауы
   setlocale(LC_ALL, "Russian");
   // str - бастапқы жол, word1 - ондағы бірінші сөз, word2 - кейінгі сөздер үшін аралық айнымалы
   string str,word1, word2; // str енгізу
   cout <<"Текстті енгізіңіз: ";
   getline (cin, str); // символдарды санауыш, басынан бастаймыз (нөлдік)
   int i = 0;   // бірінші сөзді алғанға дейін
   while (word1.empty())
     {   // егер ағымдағы таңба str - әріп
      if (isalpha(str[i]))
         // кейінгі символ әріп болғанша және str соңына жеткенше          while (i<str.size() && isalpha(str[i]))
         {  // жолдың бірінші сөзін символдық түрде құрамыз
            word1.append (1,str[i]);
            // келесі символ
            i++;
         } // әйтпесе бірден келесі символға өтеміз
      else i++;
   } // келесі сөздерді қарастырамыз.  а айнымалысы word2 сөзінің word1 сияқты таңбалардан тұратындығын білдіреді (a = true), әйтпесе a = false және word2 сөзін шығармаймыз. і санауышы қалады
   bool a;
   while (i<str.size())    // str бойынша өтеміз
   {
      if (isalpha(str[i]))    // егер әріп
      {
         a = true;   // әзірге a = true деп есептейміз
         word2.erase (0, word2.length());    // алдымен word2-ні босатамыз
         // әзірге әріптерді алудамыз және жолдың соңына жеткен жоқпыз
         while (i<str.size() && isalpha(str[i]))
```

```
        {  // word2 сөзін құрамыз
           word2.append (1,str[i]);
           // егер ағымдағы символ word1-дің ішінде табылмаса, a = false қайтарамыз
           if (word1.find(str[i])==-1)
              a = false;  // келесі әріп
           i++;  }
        // егер біз a айнымалысын false қалпына келтірмесек - яғни, word 2 сөзінің барлық символдары word1 бірінші сөзінде бар - word2-ні шығарамыз
           if (a) cout <<word2 <<endl;
        }    // келесі символ str
        else i++;
     } // Enter пернесін басқан кезде консольді жабу
     system("pause");
     // соңы
     return 0;}
```

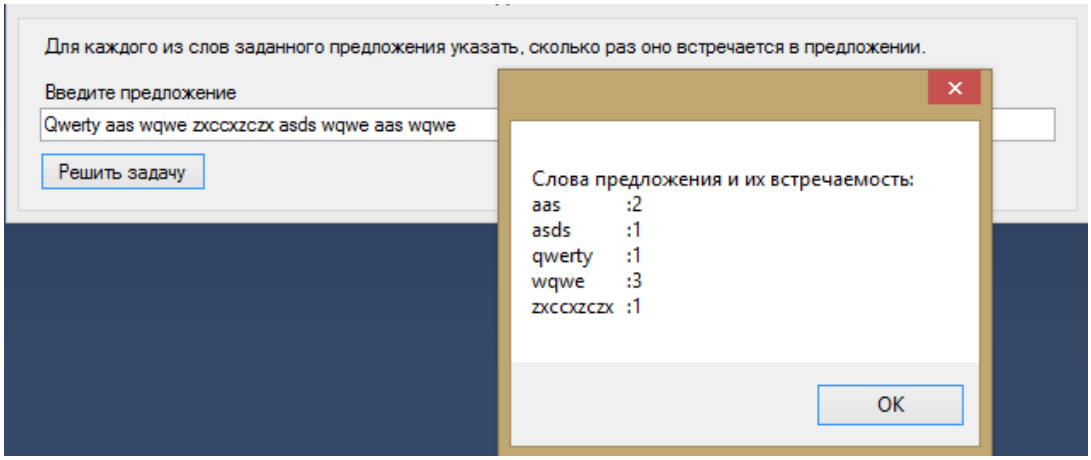


14 Берілген сөйлемдегі әрбір сөздің осы сөйлемде неше рет кездесетінін көрсетіңіз.

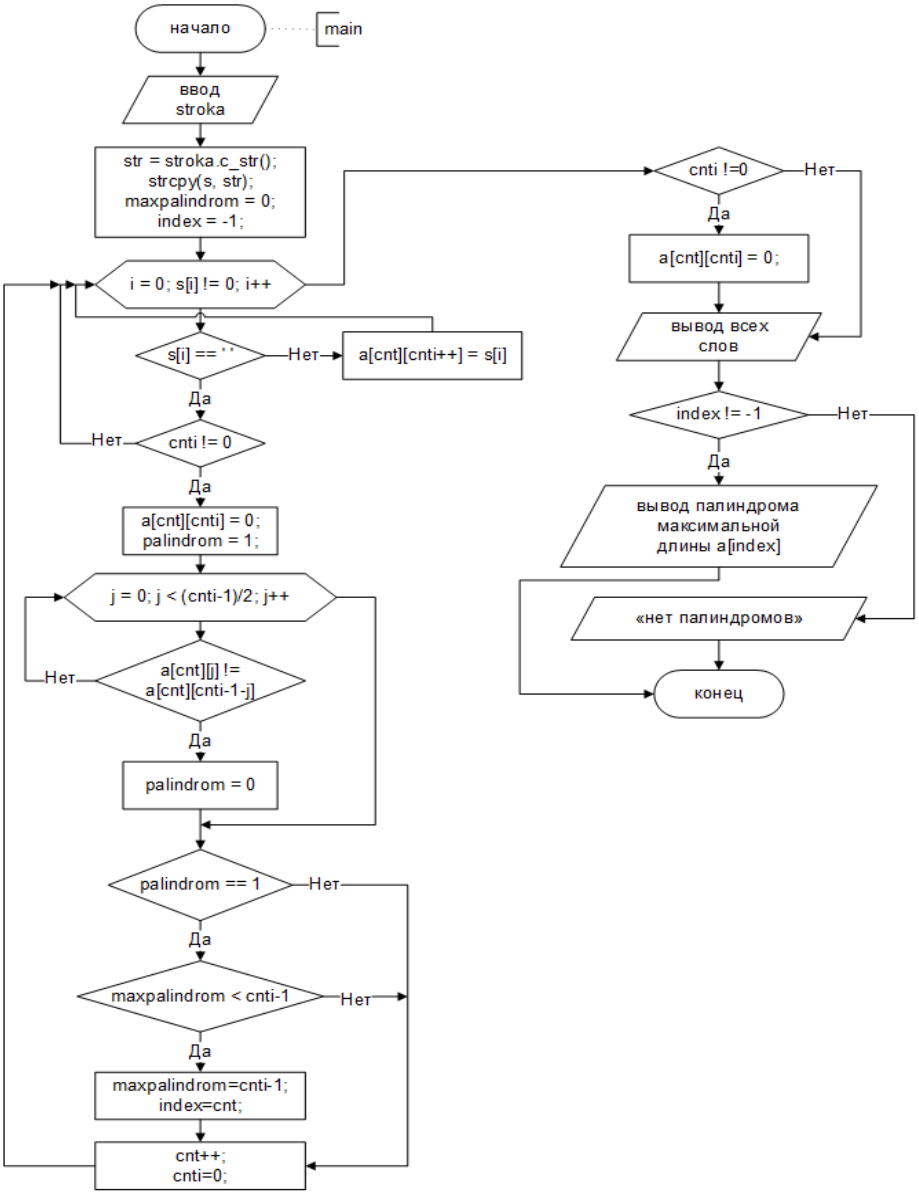



```
#include <iostream>
#include <iterator> // кіріс деректерін бөліктерге бөліп оқуға мүмкіндік беретін кітапхана (бұл жағдайда енгізу жолын сөздерге автоматты түрде бөлу)
#include <sstream>  // ағындық кіріс деректерін оқуға арналған кітапхана (бұл жағдайда жолдар)
#include <string>
#include <map>  // «Map» құрылымын қолдауға арналған кітапхана (әртүрлі типтегі жұптар, бұл жағдайда - әрбір сөз үшін нөмір береміз - мәтінде берілген сөздің кездесу саны)
 int main()
{
   // map айнымалысын жариялаймыз - "жол-сан" жұптарының массиві
   std::map<std::string, int> map;

   // line кіріс жолы
   std::string line = "enu abul mashina enu fit samolet fit";
   // кіріс жолын SS ағынына түрлендіру
   std::stringstream ss(line);

   // i - SS ағынына арналған сөз санауышы (яғни line сөйлеміндегі сөздер)
   std::istream_iterator<std::string> i = std::istream_iterator<std::string>(ss);
   // ағын бойынша өтеміз. ағымдағы сөзге сәйкес (*i):  i - санауыш,  *i - сөз. map[*i] - сан.
   for(; i != std::istream_iterator<std::string>(); ++i)
      // *i сөзін кездестіргенде – оның кездесу санын арттырамыз
```

```
        map[*i]++;

    // енді i санауышы map массиві арқылы өтеді. Әрбір map элементі үшін сөздің өзі ((*i).first) және оған сәйкес келетін сан ((*I).second) көрсетіледі
        for(std::map<std::string, int>::iterator i = map.begin();
          i != map.end(); ++i)
          std::cout << (*i).first << "\t: " << (*i).second << std::endl;
            // Enter пернесін басқан кезде консольді жабу
        system("pause");
        // соңы

        return 0;
    }
```



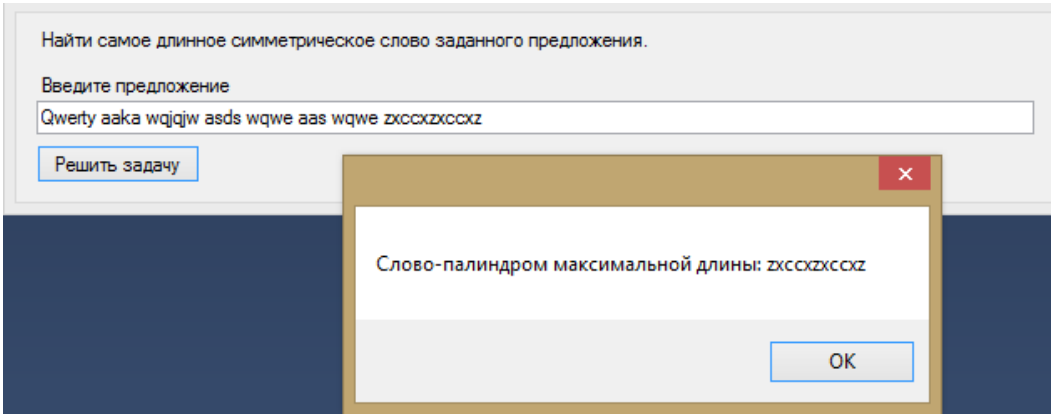
15 Берілген сөйлемнің ең ұзын симметриялық сөзін табыңыз.



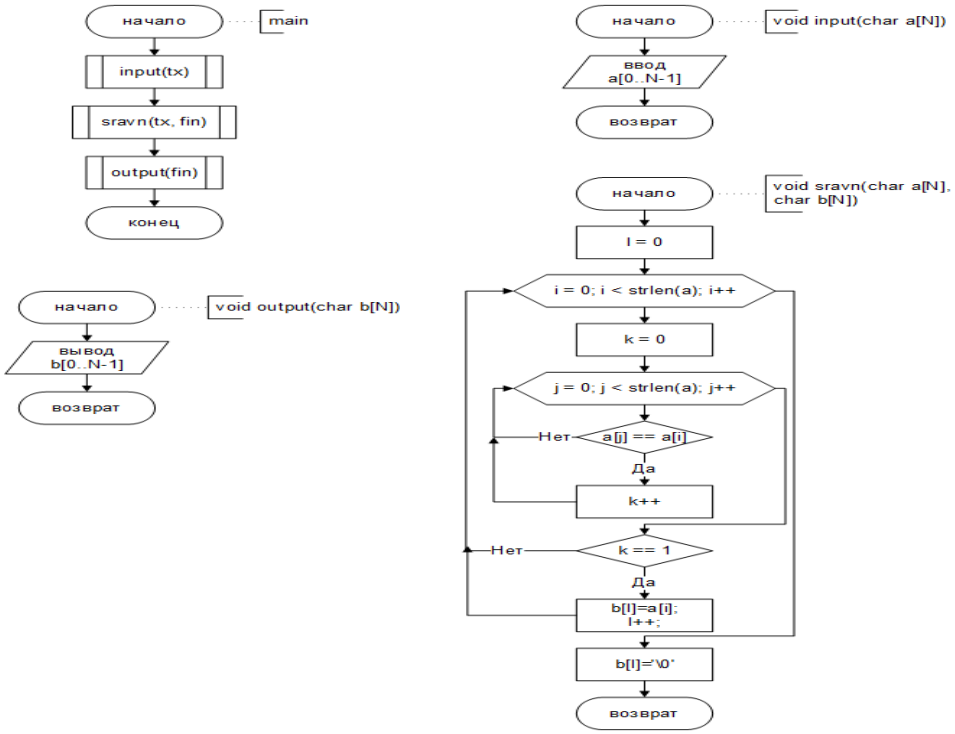

```
#include <conio.h>
#include <iostream>

using namespace std;

int main(void)
{
   // s символдар массиві – кіріс жолы (шектеу - 100 символ)
   char s[100];
   // a сөздер массиві (10 сөзден көп емес). 100 - сөздегі символдардың максималды санын көрсетеді (өйткені сөз әріптер массиві)
   char a[10][100];
   // cnt - бастапқы жолдағы сөз санауышы
```

```
int cnt=0;
// cnti - сөз ішіндегі символдар санауышы
int cnti=0;
// maxpalindrom - максималды сөз-палиндромның ұзындығын сақтайды, index – максимал ұзындықтағы сөз-полиндромның нөмірі
int i,j,palindrom,maxpalindrom,index;

// бастапқы жол
string stroka = "ssd hel55leh myym mqw2wqm world";
// str - бастапқы жолға көрсеткіш
const char *str = stroka.c_str ();
// s символдар массивіне жолды жазамыз
strcpy(s, str);

maxpalindrom = 0;
index = -1; // s жолының символдар массивін қарастырамыз
for(i= 0;s[i] != 0;i++)
{ // бос орынға тап болсаңыз - сөздің соңы
   if (s[i] == ' ')
   {      // егер сөз бос болмаса
      if (cnti!=0)
      {
         a[cnt][cnti] = 0;  // сөзді палиндромға тексеруге арналған айнымалы
         palindrom = 1;
         // соңындағы және басындағы таңбалар жұбын тексеру (палиндромды тексеру)
         for(j=0;j<(cnti-1)/2;j++)
         {
            // егер сәйкессіздіктер бар болса-сөз палиндром емес: palindrom = 0, циклдан шығамыз
            if (a[cnt][j] != a[cnt][cnti-1-j]){palindrom=0;break;}
         }  // егер сөз - палиндром: ұзындығын тексереміз. Егер үлкен болса – ұзындығын maxpalindrom-де есте сақтаймыз, index-ке - сөздің нөмірін
         if (palindrom == 1)
            if (maxpalindrom < cnti-1) {maxpalindrom=cnti-1;index=cnt;}
         // келесі сөздерге көшеміз
         cnt++;cnti=0;   }
   } // кездестірген әріптерді - a[cnt] сөзіне жазамыз
   else {a[cnt][cnti++] = s[i];}
} // соңғы сөзді "0"деп аяқтаймыз - жолдың соңы
```

```
        if (cnti !=0) {a[cnt][cnti] = 0;} // барлық сөздерді шығару
        for(i=0;i<cnt;i++)
        {printf("%s\n",a[i]);}; // a-дан index  нөырлі максималды полиндром-сөзді
шығару
        if (index != -1)
        printf("%s%s\n","Max palindrom: ",a[index]); // егер палиндромдар кездеспесе
        else printf("%s\n","No palindroms"); // Enter пернесін басқан кезде консольді
жабу
        getchar(); // соңы
        return 0;
    }
```

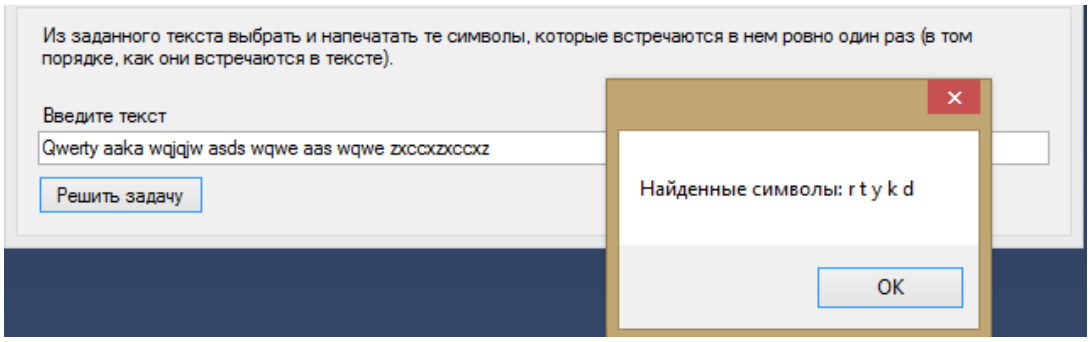


16 Берілген мәтіннің ішінен тек бір рет қана кездесетін символдарды таңдап алып, мәтінде пайда болу ретімен жазып шығыңыз.

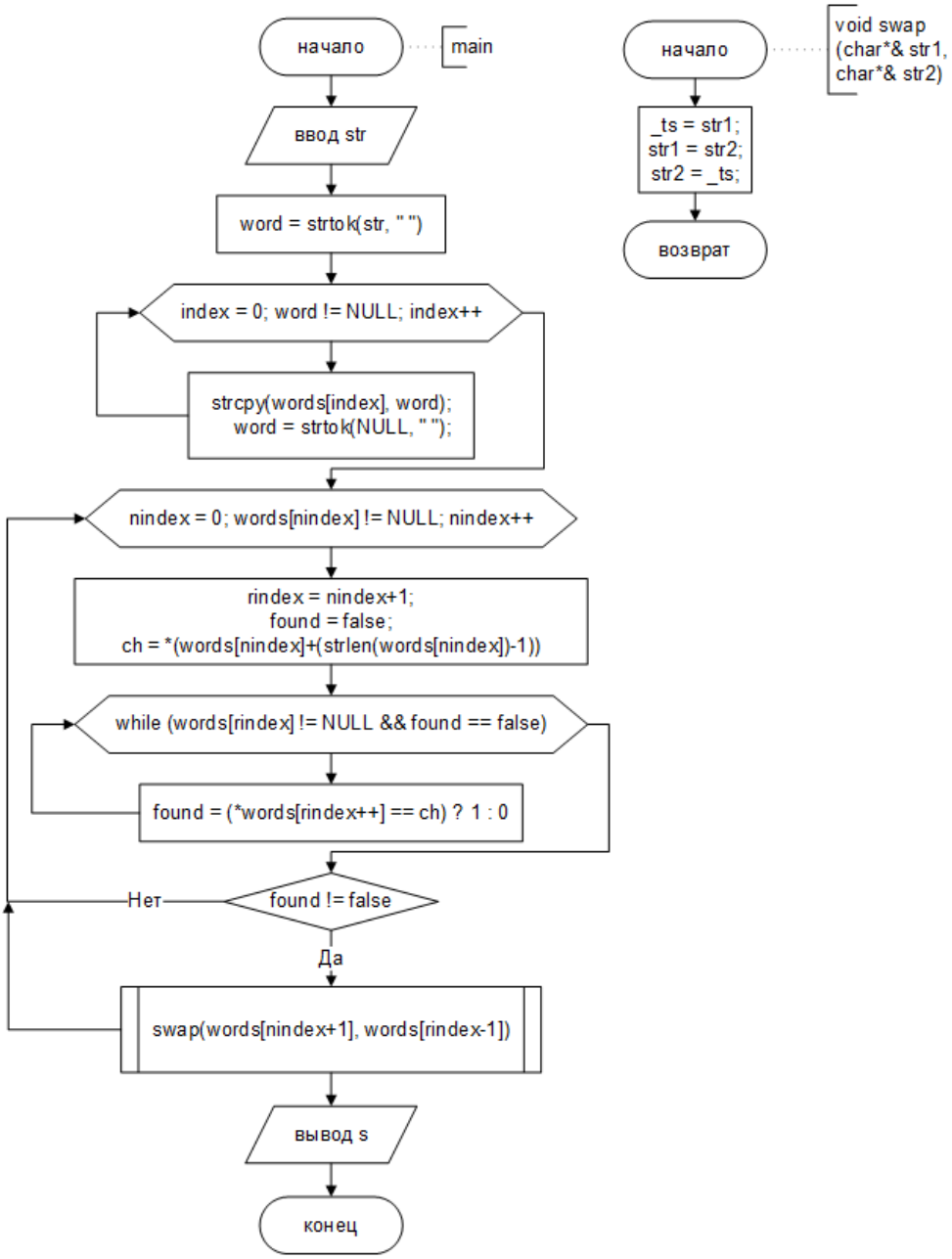


```
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <locale.h>
const int N=100;    // кіріс жолы үшін символдар санының шегі
// деректерді енгізу процедурасы
void input(char a[N])
{
   int i;  // жолдың ішіндегі символдарға арналған санауыш
   char s; // символға арналған аралық айнымалы
   printf_s("Нуктемен аякталатын текстті енгізіңіз.\n");
   // символдар консоль жолынан символ бойынша оқылады, және олар a
массивіне жазылады
   for (i=0;(s=getchar())!='.';i++) a[i]=s;
   // жол соңындағы қызметші символ
   a[i]='\0';
}
// әрбір символдың пайда болуын санау және қажетті массивті толтыру
процедурасы
void sravn(char a[N],char b[N])
{
   // k – символдардың кездесуін санаушы. l - b массивіне арналған санауыш,
оған a массивінде бір рет кездесетін сандарды жазамыз
   int k,l=0;
   unsigned int i,j;   // символдарды санауыш
   // бастапқы жолдың әрбір символын ретімен қарастырамыз
   for(i=0;i<(strlen(a));i++)
   {
      // символдың кездесуін санауышты  қалпына келтіру (0)
      k=0;
      //  массивтің барлық символдарымен рет- ретімен салыстыра отырып,
a[i] символдардың кездесу санын табамыз
      for (j=0;j<strlen(a);j++) if (a[j]==a[i]) k++;
      // егер символ тек бір рет кездесс – оны b массивіне,  l позициясына
жазамыз, санауышты келесі позицияға қоямыз
      if (k==1) {b[l]=a[i];l++;}
   }
   // b массивінің соңына қызметтік жолдың соңындағы символды жазамыз,
осылайша оны жол ретінде қарастыруға болады.
```

```
    b[l]='\0';
}
// нәтижелерді шығару процедурасы
void output(char b[N])
{
    unsigned int i; // символдар санауышы
    // нәтижелерді көрсету - яғни b массивінің символы бойынша
    printf_s("\nТекстте кездесетін бір рет пробел арқылы кездесетін символдар:\n");
    for (i=0;i<strlen(b);i++) printf_s("%c ",b[i]);
}
void main()
{
    // tx – символдар массиві (яғни, жол), символдар үшін N жад ұяшықтарын резертейміз . fin – нәтижелер массиві
    char tx[N],fin[N];
    // консольдің кириллица таңбаларын қолдауы
    setlocale(LC_CTYPE, "Russian");
    // деректерді енгізу
    input(tx);
    // есепті шешу процедурасы. Бұл процедураның ішінде tx массиві a атауымен қарастырылады, fin массиві – b ретінде
    sravn(tx,fin);
    // вывод
    output(fin);
    // Enter пернесін басқан кезде консольді жабу
    _getch();
}
```



17 Сөйлемде барлық сөздер әр түрлі әріптерден басталады. Сөйлем сөздерін (мүмкін болса) әр сөздің соңғы әрпі келесі сөздің бірінші әрпіне сәйкес келетіндей ретпен жазып шығыңыз.

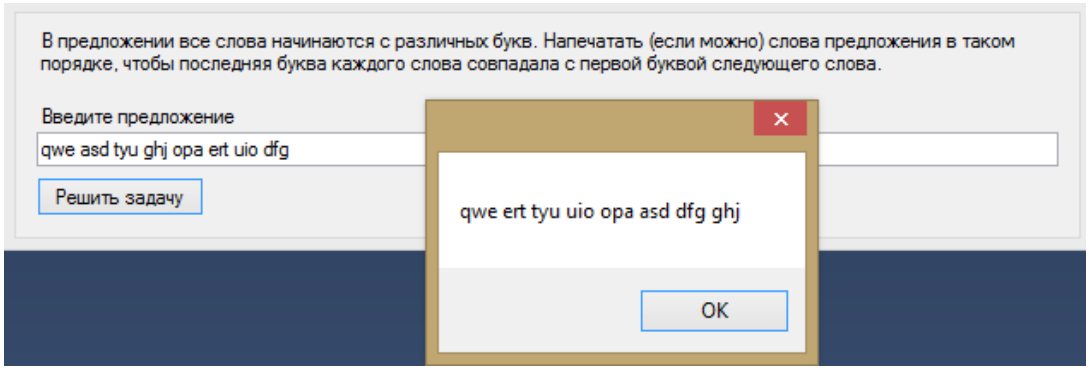


#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <memory.h>

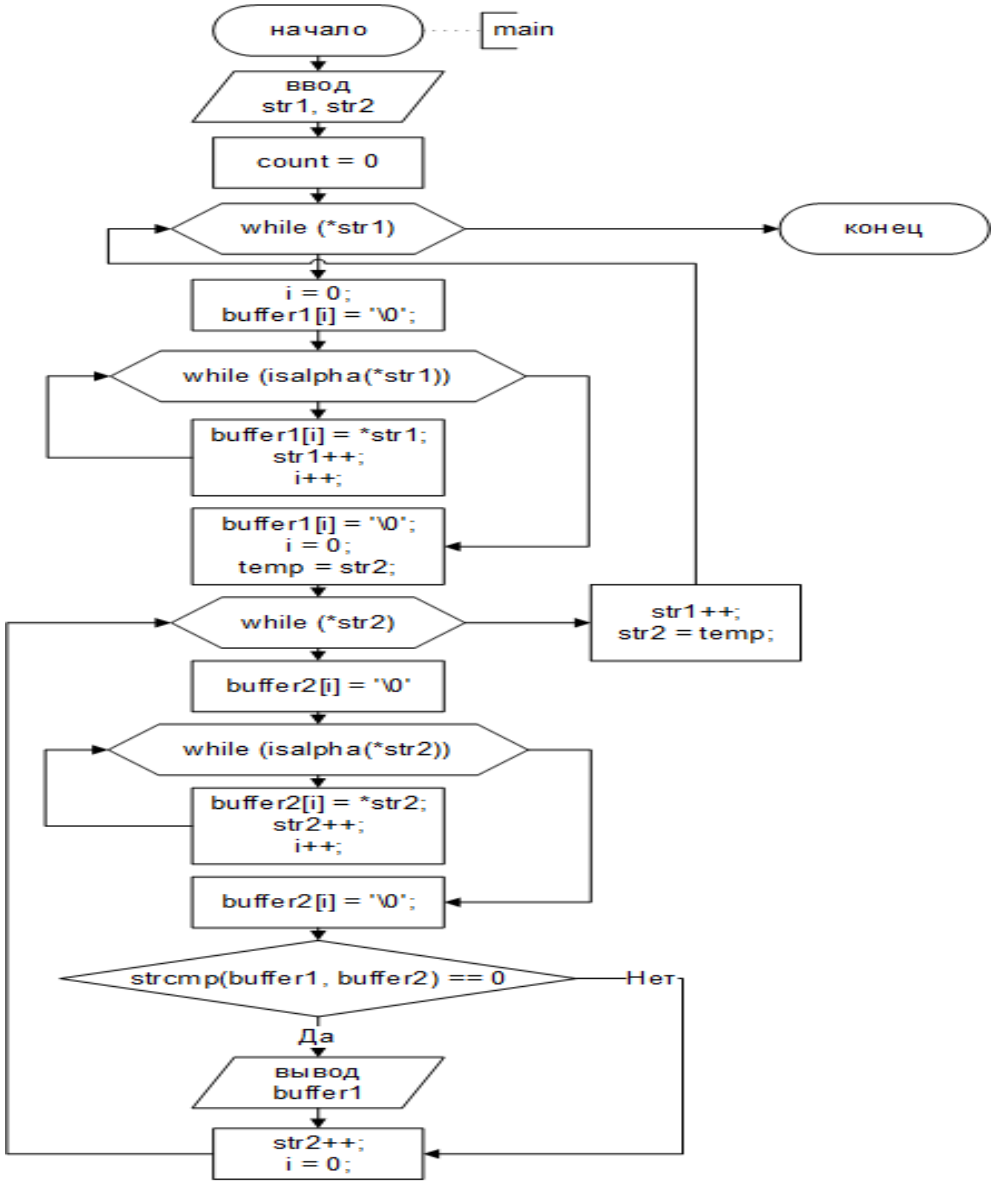
// екі айнымалының мәндерін алмастыру процедурасы (біздің жағдайда - массивтегі сөздер)
void swap(char& str1, char*& str2);*

```
int main(int argc, char* argv[])
{
   // кіріс жолы, шектеу - 256 символ
   static char str[256] = "enu tuda reshil univer letat\0";
   // оны шығарамыз
   printf("%s\n",str);

   //сөздер массиві, шектеу - 256 сөз
   char** words = new char*[256];
   // массивке орын бөлеміз
   memset((void*)words, 0x00, 4 * 256);

   // strtok процедурасы str жолын бос орынмен бөлінген ішкі жолдарға бөледі - яғни, сөздерге. word – осы сөздердің массиві    char* word = strtok(str, " ");
   // бұл сөздерді words массивіне көшіреміз
   for (int index = 0; word != NULL; index++)
   {
      // words массивінде char [] объектісін жасаймыз - бос жол
      words[index] = new char[256];
      // көшірмесін жасаймыз
      strcpy(words[index], word);
      // str жолынан өтуді жалғастыру және келесі сөзге сілтеме алу үшін, бірінші аргументтің орнына NULL жазамыз
      word = strtok(NULL, " ");
   }

   // words массивіндегі сөздерді қарастырамыз
   for (int nindex = 0; words[nindex] != NULL; nindex++)
   {
      // rindex - келесі сөздің нөмірі, found - words[nindex] сөзінің соңғы әрпінен басталатын сөзді таптық па
      int rindex = nindex+1; bool found = false;
      // strlen(words[nindex]) - сөздің ұзындығы. words[nindex] - көрсеткіш (!),егер оған минус бір сөздің ұзындығын қоссақ, біз соңғы таңбаға көрсеткіш аламыз. Яғни, ch- индексі nindex болатын сөздің соңғы символы (*символы көрсеткіштен әріп алады)
      char ch = *(words[nindex]+(strlen(words[nindex])-1));
```

```
        // енді қалған сөздерді соңына дейін қарастыра отырып, сөйлемнің
соңына жеткенше сәйкес сөздерді іздейміз. found - 1 (false = 0) болғанда – циклдан
шығамыз
        while (words[rindex] != NULL && found == false)
          // rindex++ санауышты автоматты түрде арттырады, бірақ оған
дейін осы индексі бар сөз тексеріледі - егер оның бірінші символы ch-ке тең болса
(алдыңғы сөздер тізбегінің соңғы символы) - онда found айнымалысы 1-ге
орнатылады (яғни found != false) – және циклдан шығамыз
          found = (*words[rindex++]
          == ch) ? 1 : 0;

        // егер сәйкес сөз табылса (оның индексі rindex - 1 болады, өйткені
rindex++ операциясы индексті бірден арттырды) - оны nindex + 1 орнына қойямыз
және осы жерден алынған сөзді бос орынға орнатамыз
        if (found != false)
          swap(words[nindex+1], words[rindex-1]);
      }
       // барлық сөздерді шығару
      for (int t = 0; words[t] != NULL; t++)
        printf("%s ",words[t]);
      printf("\n");

      // Enter пернесін басқан кезде консольді жабу
      _getch();
      // соңы
      return 0;
}
// екі айнымалының мәндерін ауыстыру процедурасы
void swap(char*& str1, char*& str2)
{
char* _ts = str1;
str1 = str2;
str2 = _ts;
}
```

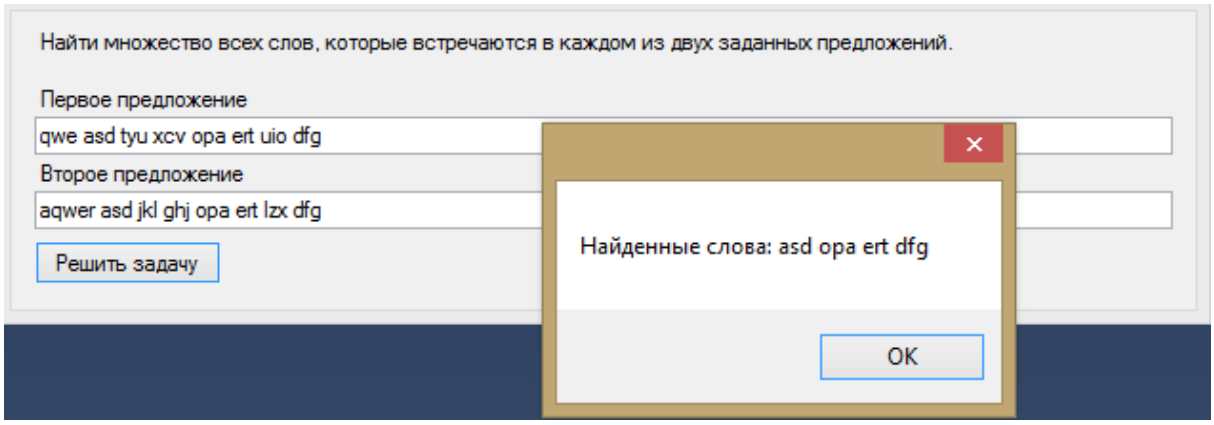


18 Берілген екі сөйлемнің әрқайсысында кездесетін барлық сөздердің жиынын табыңыз.

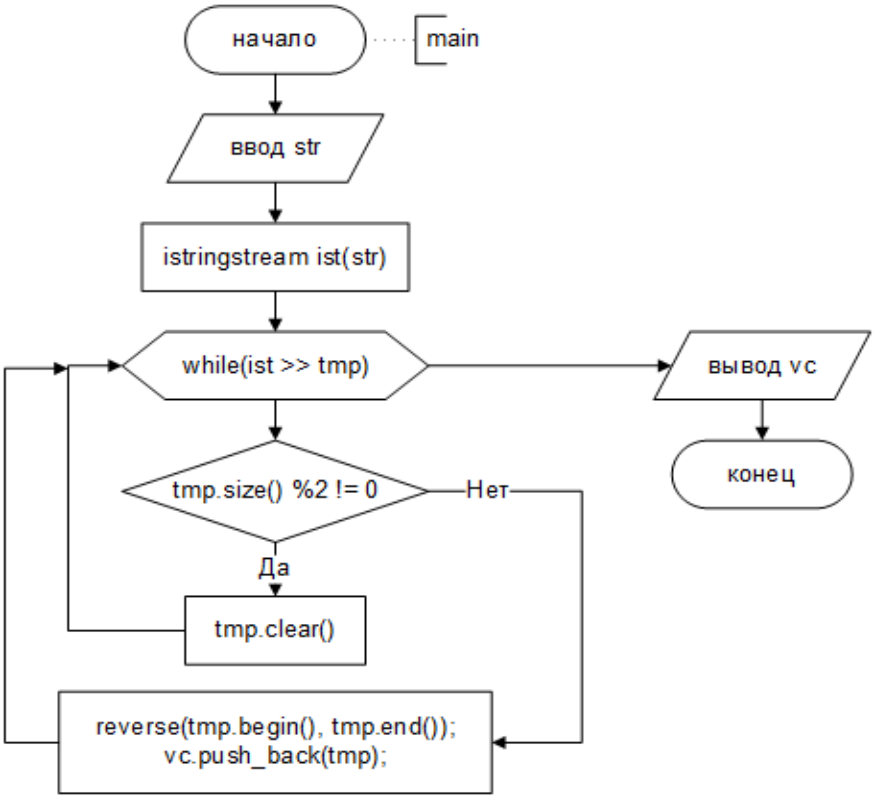


```
#include <iostream>
using std::cout;    // "std::cout" орнына жай ғана "cout" деп жазуға болады
using std::endl;    // және "std::endl" орнына"endl"
#include <cctype>
 int main(int argc, char * argv[])
   {
   // екі бастапқы сөйлем
   const char * str1 = "ENU universitet nohiditsya v Astana";
   const char * str2 = "ENU Astana FIT";
   // сөздерді жазуға арналған аралық массивтер
   char buffer1[50];
   char buffer2[50];
   int count = 0;
   // str1 және str2 – символдарға көрсеткіш болғандықтан,  оларды санауыш ретінде қолданамыз (str1++ - келесі str1 символына өту). Жолдың соңына жеткенде цикл үзіледі
   while (*str1)
      {
      int i = 0;  // сөз санауышы
      buffer1[i] = '\0';  // жол соңындағы символ - бос сөз
      // әзірге str1 әріптерін аламыз
      while (isalpha(*str1))
         {
         // әріпті buffer сөзіне жазамыз, санауышты арттырамыз
         buffer1[i] = *str1;
         str1++;
         i++;
         }   // әріп емес болып табылатын символды алған кезде циклден шығамыз - яғни, сөздің соңы
      // сөзді жолдың соңымен аяқтаймыз
      buffer1[i] = '\0';
      i = 0;       // екінші сөйлемді temp айнымалысына сақтаймыз (содан кейін str2 көрсеткішін temp - ге орнатамыз-сөйлемді қалпына келтіреміз)
      const char * temp = str2;
      // екінші сөйлемнің соңына жеткенше
      while (*str2)
         {   // сол сияқты екінші сөйлемнің сөздерін қарастырамыз
         buffer2[i] = '\0';
         while (isalpha(*str2))
```

```
        {
        buffer2[i] = *str2;
        str2++;
        i++;
        }
    buffer2[i] = '\0';   // екі сөйлемнен алынған екі сөзді салыстыру. strcmp = 0 - сөздер сәйкес келгенде
    if (strcmp(buffer1, buffer2) == 0)       // ізделінді сөзді консольге шығарамыз
        cout << buffer1 << endl;  // екінші сөйлем бойынша жалғастырамыз
    str2++;
    i = 0;
    }  // екінші сөйлемді қарастырып болған соң - str2 көрсеткішін қайтадан басына қойямыз, str1 сөйлемінде келесі символға (яғни келесі сөзге) жылжимыз
    str1++;
    str2 = temp;
    } // Enter пернесін басқан кезде консольді жабу
system("pause");  // соңы
return 0; }
```

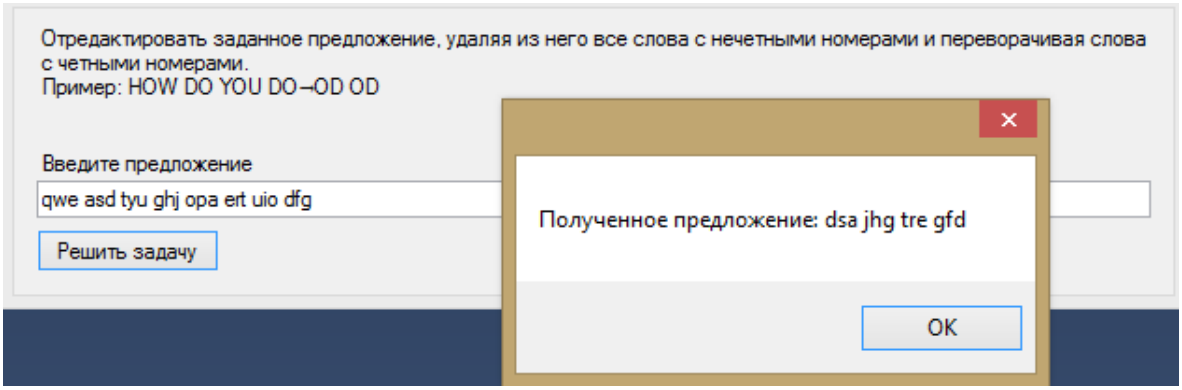


19 Берілген сөйлемді одан нөмірі тақ болатын барлық сөздерді алып тастап, және нөмірі жұп болатын сөздерді кері аудару арқылы өңдеңіз. Мысалы: HOW DO YOU DO→OD OD

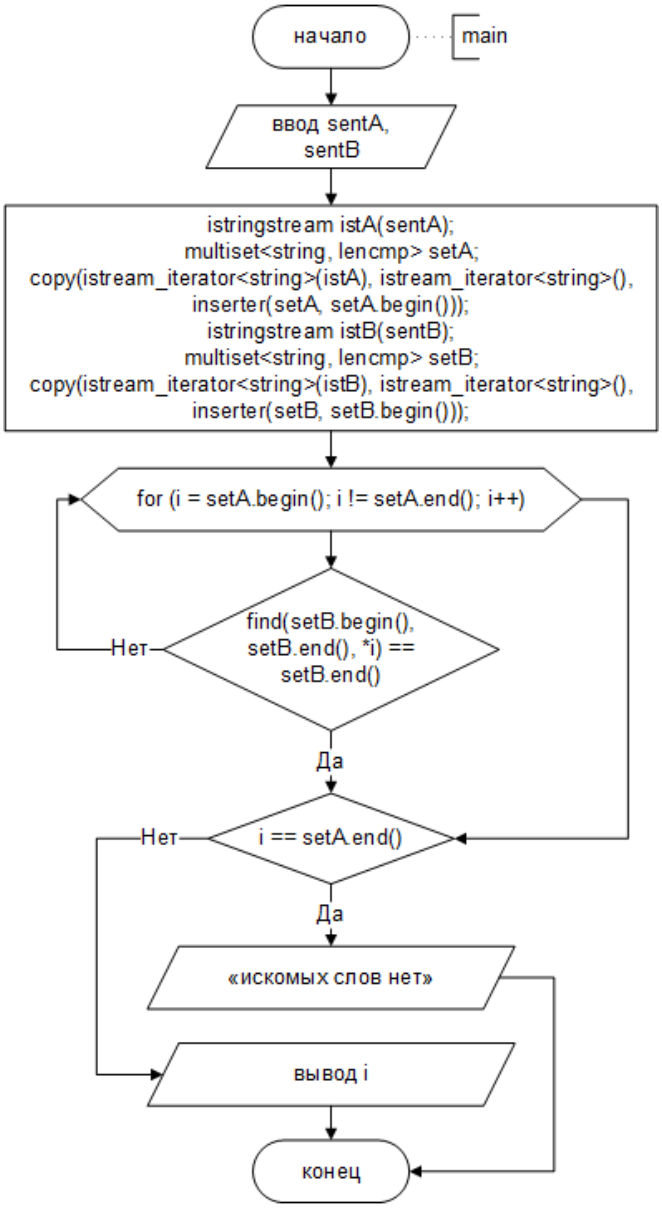



```
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <algorithm> // кітапханада символдарын кері ретімен жазылған жолды алу үшін reverse1 процедурасы бар
using namespace std;

int main()
{
   // консольдің кириллица таңбаларын қолдауы
   setlocale(LC_ALL, "Russian");
   // str - бастапқы жол
   string str;
   // консольден str енгізу
   cout<<"Текстті енгізіңіз\n";
   getline(cin, str);
   // бір сөзді оқуға арналған аралық айнымалы
```

```
    string tmp;
    // str сөйлемін сөзбен оқу үшін ist ағынын жасаймыз
    istringstream ist(str);
    // вектор (массив) vc-нәтиже сөздерін сақтайды
    vector<string> vc;
    // tmp-де ist ағынынан бір-бір сөз жазылады (яғни str сөйлемінен)
    while(ist >> tmp)
    {
       // ұзындығы тақ  болатын сөз - сөздің айнымалысын тазалау
       if(tmp.size() %2 != 0)
         tmp.clear();
       else
       // әйтпесе - кері айналдыру және vc массивіне көшіру
       {
         reverse(tmp.begin(), tmp.end());
         vc.push_back(tmp);
       }
    }
    // vc массивіндегі сөздерді шығарамыз
    for(vector<string>::iterator it=vc.begin(); it != vc.end(); ++it)
      cout<<*it<<' ';
    cout<<'\n';
    // Enter пернесін басқан кезде консольді жабу
    system("pause");
    // соңы
    return 0;
}
```



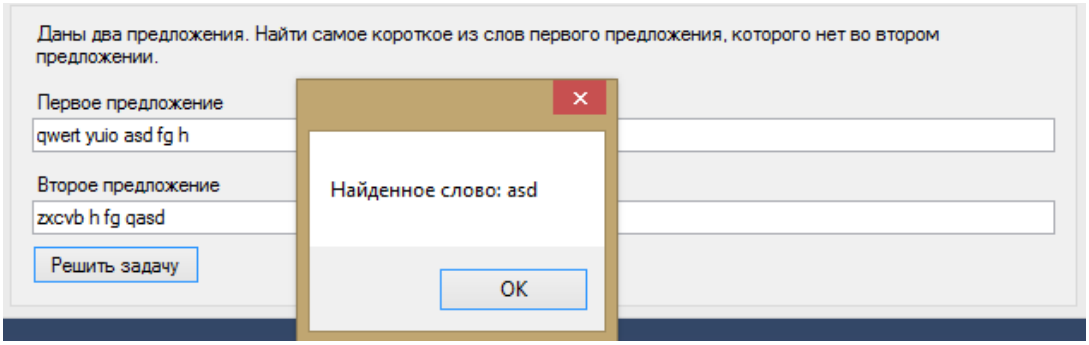
20 Екі сөйлем берілген. Бірінші сөйлемдегі екінші сөйлемде кездеспейтін ең қысқа сөзді табыңыз.




```
#include <iostream>
#include <string>
#include <sstream>
#include <set>  // multiset жиындарымен жұмыс істеуге арналған кітапхана
#include <algorithm>
#include <iterator>

// lencmp құрылымы атрибуттары жоқ, бірақ жолдың ұзындығын салыстыратын операторды қамтиды. Сөздердің жиынындағы (multiset) сөздерді ұзындықтарының өсуі бойынша сұрыптау үшін қолданылады
```

```
struct lencmp {
  bool operator () ( const std::string & a, const std::string & b ) {
    return a.size() < b.size();
  }
};

int main(){  // sentA, sentB – бастапқы сөйлемдер
  std::string sentA, sentB, tmp;  // консольден екі сөйлемді де оқимыз
  std::cout << "Birinshi soilem: ";
  std::getline(std::cin, sentA);
  std::cout << "Ekinshi soilem: ";
  std::getline(std::cin, sentB);

  // sentA  сөйлеміндегі сөздерді ағынмен оқу – istA ағыны
  std::istringstream istA(sentA);
  // (multiset) setA жиынтығын жариялаймыз, оған қосылған кезде сөздер lencmp операторына сәйкес автоматты түрде сұрыпталады - яғни, ұзындықтарының өсуі бойынша (ең қысқасы - жиынтықтың басында)
  std::multiset<std::string, lencmp> setA;
  // біз бұл жиынға istA ағынынан сөздерді жазамыз (яғни, sentA сөйлемінен). Бірінші аргумент - ista ағынының басталу итераторы, екіншісі - бос итератор (яғни, ағынның соңы), үшіншісі - жинақтармен жұмыс істеуге арналған арнайы inserter түрі, begin сөзі басынан бастап жинаққа жазуды көрсетеді
  std::copy(std::istream_iterator<std::string>(istA),
std::istream_iterator<std::string>(),
    std::inserter(setA, setA.begin()));
  // сыған ұқсас sentB жолы үшін
  std::istringstream istB(sentB);
  std::multiset<std::string, lencmp> setB;
  std::copy(std::istream_iterator<std::string>(istB),
std::istream_iterator<std::string>(),
    std::inserter(setB, setB.begin()));
  // i - setA жиынтығы арқылы өту үшін көмекші айнымалы
  std::multiset<std::string, lencmp>::const_iterator i;
  for ( i = setA.begin(); i != setA.end(); ++i )
    // *i - сөз көрсеткіші. Егер бұл сөз табылмаса (теңдік орындалады) – циклдан шығамыз. SetA жиынтығы ұзындықтың өсуі бойынша сұрыпталғандықтан, циклден шыққан кезде * I көрсеткіші strB-де жоқ strA-дағы ең қысқа сөзді көрсетеді
    if ( std::find(setB.begin(),
    setB.end(), *i) == setB.end() )
```

```
            break;
        // setA арқылы өту соңына дейін аяқталса - циклден шығу орындалған жоқ - демек strA-дағы барлық сөздер strB-ның ішінде бар және тапсырманың шешімі жоқ дегенді білдіреді
        if ( i == setA.end() )
            std::cout << "Soilemde ozgeshe sozder zhok!" << std::endl;
        else // әйтпесе ізделінді сөзді шығарамыз
            std::cout << "Ekinshi soilemde zhok en kiska soz " << *i << std::endl;
        // Enter пернесін басқан кезде консольді жабу
        system("pause")    // соңы
        return 0;
    }
```



3 Программалау бойынша оқыту нәтижелерін бағалау критерийлері

Жалпы оқыту үдерісі оқытушы мен білім алушының бірлескен әрекетінен тұрады. Оқытушы мен білім алушының өзара қарым-қатынасы бірқатар түбегейлі әр түрлі іс-қимылдарды қамтитынын көрсетеді: ақпараттық (білім алушыға қандай да бір түрде оқу материалы беріледі: дәрістер, мәтіндер, суреттер, графиктер және т.б.); жаттықтыратын (білім алушы есептерді шеше отырып білімін тәжірибеде пайдалануға үйренеді); бақылайтын (білім алушылардың білімдері тексеріледі және бағаланады); әкімшілік (әрбір білім алушының оқыту үдерісі көрсеткіштерінің есебі жүргізіледі). Әрине, кез-келген пәндік саланы зерттегенде білім алушылар үшін алғашқы екі іс-шара түрі негізгі болып табылады. Бақылау және басқару оқу үдерісін басқаруға және кері байланысты қамтамасыз етуге мүмкіндік береді. Әрекеттің бұл екі түрін үнемі оқытушы атқарады. Программалық жасақтама технологияларын (ПҚ) оқытудың ерекшелігі білім алушының біліктілік пен дағдыларын қалыптастыруға бағытталған жаттықтыратын оқыту болып табылады. Мысалы, программалауды үйрету барысында білім алушылар зертханалық және курстық жұмыстарды орындай отырып, өте көп программалар жасаулары қажет. Жобалау әдістерін оқып-үйрену барысында курстың көп бөлігін зертханалық жұмыстар құрайды, оның барысында білім алушылар UML тілін қолдана отырып, модельдерді құру тәжірибесін жинақтайды. Осылайша, ПҚ дайындау технологияларын оқыту кезінде шығармашылық тәсіл басымдылыққа ие. Оның үстіне бақылаудың рөлі едәуір артады, оның қызметі білім алушының білім және біліктілік деңгейін бағалау болып табылады. Осы бағалаудың негізінде білім беру үдерісін бейімді басқару жүзеге асырылады. Дегенмен, ПҚ дайындау технологияларын оқыту кезінде бағалау өте күрделі болып келеді. Нақтылық үшін программалауды оқытудағы бағалау мәселесіне тоқталған жөн. Гуманитарлық ғылымдар мен көптеген жаратылыстану ғылымдарын оқытудағы білім деңгейін бағалау үшін кеңінен қолданылатын тестілеу технологиялары программалауды оқытуда жеткіліксіз болып табылады. Себебі олар білім алушылардың программаларды әзірлеу қабілетін бағалауға мүмкіндік бермейді. Сондықтан, программаны бағалау бейресми үдеріс болып табылады және қойылатын баға негізінен субъективті болып табылады. Бағалауға әсер ететін субъективті факторлардың қатарына оқытушының программалаудың белгілі бір стиліне, нәтижелерді ұсыну тәсіліне, программалық интерфейстің ыңғайлығына байланысты болуы мүмкін. Сондай-ақ, оқытушы қорытынды бағаны қою барысында білім алушының күтілетін білім деңгейіне жиі бағдарланады. Бағаға оқытушы мен білім алушының арасындағы тұлғааралық қарым-қатынастарының әсері болмай қоймайды. Осылайша, дәстүрлі оқыту кезінде дайындалған программалар үшін қойылған бағаның обьективтілігі туралы сөз етуге болмайды.

Сондықтан білім алушының жұмысын объективті бағалау - автоматтандырылған оқыту жүйесінде шешілетін өзекті міндет.

Оқыту жүйесі екі бағаны анықтауы қажет: нақты тапсырманы орындағаны үшін қойылатын бір реткі баға және білім алушының білім деңгейінің ағымдағы бағасы. Екінші баға оқу деңгейін анықтайды, ол пәндік саланы зерттеудің траекторияларын түзетуге арналған бейімделуші алгоритмдермен де қолданылады. Оқу деңгейін бағалау тек тәжірибелік тапсырмаларды іске асыруға ғана байланысты екендігін атап өту керек, теориялық материалды зерттеу оқыту деңгейіне әсер етпейді. Әрбір нақты тапсырманы орындауда жүйе білім алушының білім моделін түзетеді, оның негізінде оқыту деңгейі анықталады. Белгілі тапсырманы орындауы үшін бір реттік бағаның есептелуін қарастырайық. Есептеу үшін кортеж ретінде қарастырылатын тапсырма моделі қолданылады:

$$Me = < T, Mm, MS, L, w >$$

мұнда T – тапсырма мәтіні;

Mm – программаны тестілеу моделі;

MS – программаның эталонды статикалық моделі;

$L \in [0, 1]$ – тапсырма күрделігінің деңгейі; тиісті тапсырманы іріктеу үшін бейімделуші алгоритмдермен қолданылады.

$w \in [0, 1]$ – берілген тапсырма «байланған» білім бірліктерін меңгеруге арналған тапсырма салмағы (маңыздылығы). О(е) тапсырмасын орындағаны үшін қойылатын бір реттік баға тапсырма моделі негізінде есептеледі (1):

$$О(е) = (Ов + Ок) \times w \quad (1)$$

мұнда Ов – программаны орындауының бағасы;

Ок – программа сапасының бағасы. Орындау үшін баға тестілеу моделі бойынша қойылады. Мт программасын тестілеу моделі үш жиынтықты құрайды:

1. $Tt = \{ t1, t2, \ldots, tk \}$ – орындалуды тексеруге арналған тестілер жиынтығы.

2. $Rt = \{ r1, r2, \ldots, rk \}$ – Tt тестілер жиынтығының эталонды жауаптардың жиынтығы.

3. $Dt = \{ d1, d2, \ldots, dk \}$ – диагностикалық хабарламалардың жиынтығы.

Тестілеу моделін жүйе енгізілген программаның дұрыстығын автоматты түрде тексеру үшін қолданады. Тесті орындау нәтижесі r(ti) $ti \in Tt$ тиісті эталонды нәтижемен $ri \in R$ салыстырылады. Салыстыру нәтижелері B бульдік векторын құрайды. Орындау үшін баға формула бойынша есептеледі, мұнда *bi* – вектор элементі 0 немесе 1 тең. Осылайша, егер барлық тестілер өтілсе, онда орындау бағасы максималды болады және 0,6 тең. Егер қандай да бір тестілер өтілмесе, баға азаяды. *di* диагностикалық хабарлама (түсініктеме) білім алушыға $r(ti) \neq ri$ болған жағдайда беріледі. *Ок* сапа бағасы

Ов = max болған жағдайда ғана есептелінеді. Программа сапасының бағасын есептеу үшін MS программасының статикалық моделі пайдаланылады, бұл соңғы бірізділікті білдіреді:

$$MS = MI1, MI2, \ldots, MIn$$

мұнда $MIj,\ 1 \le j \le n$ – нұсқаулықтың статикалық моделі.

Жалпы айтқанда, нұсқаулықтың статикалық моделдері программалау тілдеріне тәуелді. Әйтседе программалаудың барлық тілдерінде айнымалыны және тұрақтыларды хабарлаудың, массивтерді хабарлаудың, тағайындау операторларының, екі түрдің шартты операторларының, цикл операторлардың үш түрінің, шақыру функциясы операторының, жаңа функцияларды (кіші бағдарламалар) анықтайтын құралдардың нұсқаулықтары қоса беріледі. Көптеген нұсқаулықтарда нұсқаулықтардың реті қоса берілуі мүмкін (шартты оператор, цикл операторы, функцияларды анықтау). Аталған нұсқаулықтардың формалды статикалық моделдерін қарастырайық. Әрбір модель кортежді құрайды. Айнымалыны хабарлау моделі:

$Mvar = < T, V [, E] >$ түрінде болады

мұнда T – айнымалының типі;
V – айнымалының атауы;
E – өрнек.

Квадрат жақшалар модель элементінің болмауы мүмкін екенін көрсетеді. Тұрақтыны хабарлау моделінің түрі:

$$Mconst = < T, V, E >$$

мұнда өрнек міндетті болып табылады.

Массивті хабарлау моделі төмендегідей берілуі мүмкін:

$$Marray = < T, V, S(E) >$$

мұнда S(E) – массивтің өлшемін анықтайтын өрнектердің реті.

Операторлардың статикалық модельдерін төмендегідей анықтайды:

$Mset = < V, E >$ – тағайындау операторы;

$$Mif = < E, MS(t) [, MS(f)] >$$ – if операторы

мұнда MS(t) және MS(f) – then және else бұтақтарына сәйкес келетін қоса берілген нұсқаулықтардың реттері. else бұтағы (MS(f) реттілігі) болмауы мүмкін;

$Mwhile = < E, MS >$ – алғы шарты бар цикл;

$Mdowhile = < E, MS >$ – кейінгі шарты бар цикл;

Моделдер тек атауларымен ерекшеленеді. MS – бұл циклдің денесі.

Mfor = < E1, E2, E3, MS > – санауышы бар цикл;

Mcall = < F, S(E) > – функцияны шақыру, мұнда F – функция атауы; S(E) – функция аргументтері өрнектерінің тізімі.

Mfunction = < Mheader, MS > – функцияны анықтау моделі, мұнда Mheader – функция тақырыбының моделі, ал MS – функция денесі. Функция тақырыбының моделінің (прототипінің) түрі:

Mheader = < F, T, MSdecl >

мұнда F – функцияның аты;

E – функцияның қайтатын мәнінің түрі;

MSdecl – параметрлерді хабарлау реті;

MSdecl өзіне тек Mvar, Mconst, Marray хабарлау модельдерін жатқызады. Осындай модельдерді кез келген программалау тілі үшін анықтауға болады.

MS программасының статикалық моделі ерекше түрдегі тамырлы бинарлы ағаш болып табылады. Нұсқаулықтар модельдері шыңдар болып келеді. Әрбір шың, соңғысынан басқа, келесі шың next-доғамен байланысты. Осылайша, программа нұсқаулықтарының реті next-доғамен байланысты шыңдардың ретін көрсетеді. Егер нұсқаулықтың статикалық моделінде MS элементі болса, онда ағаштың тиісті шыңының ұрпағы болады, онымен child-доғасы байланысты. Осылайша, child-доғасы бар әрбір шыңның өзі осы сияқты түрдің еншілес ағаштың тамыры болып келеді, сондай-ақ иерархия деңгейінің сандары шектелмеген.

Қарапайым мысал қарастырайық:

int max = a[0];	*Mvar*
int k = 0;	*Mvar*
for(int i = 1; i < 10; ++i)	*Mfor _ MS =*
{ if (a[i] > max)	*Mif _ MS(t) =*
{ max = a[i]; k = i; }	*Mvar*
}	*Mvar*
Printf (“max = %d”, max);	*Mcall*

Сол жақта - массивтің максималды элементін есептеуге арналған қарапайым цикл (массивтің хабарлауы көрсетілмеген), оң жағында - программаның статикалық моделінің сызбасы ұсынылған. Сызбада тек child-байланысы көрсетілген. Бірінші баған ағаш шыңдарының негізгі реттерін көрсетеді:

MS = Mvar, Mvar, Mfor, Mcall

For оператор циклінің цикл денесі болады, сондықтан Mfor шыңы MS-тің *child*-доғамен байланысты. Шыңдардың бұл реттілігі тек *Mif*-те болады.

Шартты оператордың then-бұтағы бар, сондықтан *Mif* шыңы *MS(t)* тің *child*-доғасымен байланысты, ол *Mvar, Mvar* шыңдарының реттілігін көрсетеді.

Программаның эталонды статикалық моделін оқытушы жасайды.

Оқу жүйесі білім алушының жұмысты орындау кезінде MS ұқсас статикалық моделін жасайды. Программаның сапасын бағалау осы екі модельді салыстыру кезінде есептеледі. Жасалған статистикалық және эталондық модельдерді ағаштарды салыстыруға әкеледі. Жалпы алғанда, екі графикті салыстыру мәселесі NP-толық болып табылады, бірақ біздің жағдайда статистикалық модельдер кішігірім ағаштар болып табылады, сондықтан салыстыру эвристикалық рекурсивті толық іздеу алгоритмімен орындалуы мүмкін. Ок бағалау білім алушының статикалық моделі эталонды статикалық модельдің D = f(MS, Ms) «қашықтығы» ретінде есептелінеді. Қашықтық диапазонға [0; 0,4] әкеледі, сондықтан жұмыстың жалпы нәтижесі 1-тен аспайды. Нөлдік қашықтық 0,4 мәніне ауысады.

Білімнің моделі келесі білім векторынан тұрады:

$$K(s) = (p1, \ldots, pn)$$

мында n = |V| – білім бірлігінің саны білім моделінде;

pi = p(vi|Ai), 1 £ i £ n – шартты мүмкіндік, білім алушының тәжірибелік тапсырмаларды орындау барысында жинақталған жүйелер *Ai* көрсеткіштерінде *vi* білім бірліктерін меңгерген *s* білім алушының мүмкіндіктерін көрсетеді. Нақты тапсырмаларды орындау барасындаа Î A жеке көрсеткіші болып O(e) бағасы табылады.

Осылайша K (*s*) векторының әрбір *pi* элементі *s* білім алушының *vi* білім бірлігін білу дәрежесін білдіреді. Белгілі болғандай, оқу материалын меңгерудің бес деңгейі ажыратылады: түсіну, тану, жаңғырту, қолдану, шығармашылық әрекет. Білімнің әрбір бірлігі үшін меңгеру деңгейі ретінде *vi* білім бірлігін білім алушының білім деңгейіне тиістісіне қоюға болады: білімі жоқ (A), жаңадан бастаушы (*N*), бастаушы (B), озық (*F*), сарапшы (E). Осылайша, білім бірлігі *A, N, B, F, E* мәні бар дискретті кездейсоқ шама ретінде түсіндірілуі мүмкін. Әрбір білім бірлігі үшін (кездейсоқ шамадағы) ықтималдықтың үлестірілуін есептеу үшін байестік сенім желісі пайдаланылады. Желі жоғарыда анықталған W қатынасының циклдік емес сызбасы болып табылады. Кейбір мәселені білім деңгейінің бастапқы ықтималдылығын анықтау құрайды. Ол үшін білім алушыларға тестілер жиынын ұсынады, оларды орындай отырып білім алушы жаңадан бастаушы деңгейіне жетеді. Жаңадан бастаушы деңгейінен бастап, білім алушы программалауға күрделі емес тапсырмалардан күрделірек тапсырмалар алады. *L* тапсырмасының күрделілік деңгейі білімнің келесі білім деңгейлеріне сәйкес келеді:

1. *L* £ 0,4 – жаңадан бастаушы (*N*).
2. 0,41 £ *L* £ 0,7 – бастаушы (*B*).

3. 0,71 £ *L* £ 0,9 – озық (*F*).

4. 0,91 £ *L* £ 1 – сарапшы үшін (*E*).

Білім векторы негізінде білім алушылардың интегралды *I*(*s*) деңгейі анықталады. Уақыттың әрбір нақты кезеңінде *K*(*s*) векторының әртүрлі элементтері білімнің әр басқа деңгейін көрсетеді: білімнің жоқтығынан сарапшыға дейін. Оқудың сабақтастық деңгейі тиісті деңгейдегі *K*(*s*) векторының элементтеріне сәйкес есептеледі. Жалпы ұстаным мынадай: *L* (*L* = *N, B, F, E*) деңгейіне жеткен болып есептеледі, егер *L* деңгейінің элементтері білім векторында 70%-дан аз болмаса, қалған 30 %-ды алдыңғы деңгей құрайды.

Программалардың орындалуының дұрыстығын бағалау қарапайым, бірақ жеткіліксіз. Программаны бағалаудың обьективтілігін жоғарылату программалық кодтың сапасын ескеруді қажет етеді.

Оқу жетістіктерін бағалаудың бұлжытпас алғышарты - тексеру толыққанды және жеткілікті болу керек. Ол үшін программалық қамтамасыздандырудың өмірлік циклін толық қамтитын тексеру материалдары дайындалуы қажет. Есептің қойылымынан бастап, енгізу кезеңіне дейінгі қандай жұмыс жүреді, есептің қойылымы бар, алгоритм, блок схема, псевдокодтарын қандай да ортада кодтайтын программалық өнімді тестілеу жүргізуге болады, ол ендіруге дейін қамтылуы қажет. Сол кездегі іс-әрекеттік негізде ол бірінші, екінші түрі жеке тұлғаға бағытталған оқу үдерісі жүру керек. Бағалаған кезде әрбір кезеңде алған білімдерін бағалау іс-әрекеттік деңгейде зерттеп, кешенді бағалау керек

Болашақ информатика мамандарын визуалды программалау бойынша бақылау-бағалау материалдарының мазмұны программалаудың визуалды тілдерін бағалау критерийлерін, сондай-ақ оқыту нәтижелерін бағалаудың әдістемелік қағидаларын қанағаттандыруы тиіс.

Программалаудың визуалды тілдерін бағалау критерийлері бақылау-бағалау материалдарына мынадай талаптар қояды:

- қабылдау визуалдылығы (Visual Nature);
- функционалдылық (Functionality);
- түсіну жеңілділігі (Ease of Comprehension);
- парадигма қолдауы (Paradigm Support);
- масштабтылық (Scalability).

Визуалды программалау бойынша бақылау-бағалау материалдарына қабылдау визуалдылығының критерийлерін *(Visual Nature)* ұстануды қамтамасыз ететін тапсырмалар болған жөн.

Программалаудың кез келген тілін меңгеруде болашақ информатика мамандары визуалды елестету сипаттамасын және программалау нысанның визуалдылығындағы негізгі компоненттерді қолдану тәсілдерін білулері қажет. Аталмыш контекстегі визуалды термині графиктер, диаграммалар мен белгілерді білдіреді. Мәтіндік елестетулердің визуализациясының әдістері, программалауды меңгеру тілінің синтаксисі мен семантикасының визуалдылық

зерттеуі, тиімді тәсілдермен тағы сол сияқты, кеңістікте орналасуды қолдану бұл критерийдің бөлшегі болып табылады. Олардың аз болса да басқа формалары бар. Студенттер Visual Basic графикалық интерфейстерді қалыптастыруда қолдана алатындығын түсінулері керек, бірақ ол визуалды тіл болып табылады. Шынында, Visual Basic визуалды аспектіде өте шектеулі.

Визуалды программалаудың функционалдылығы (Functionality) мәтіндік программалық модульдерді біріктіруде визуалды манипуляцияларды қолдануда болашақ информатика мамандарының құзіреттілігін бағалау арқылы тексеріледі. «Визуалды программалау тек жоғары деңгейде орындалады». Сондықтан визуалды программалау саласының болашақ информатика мамандарын Тьюринг машинасы бойынша білімін, C, ADA, Пролог и C++, C# программалау тілдерін Visual Studio орталарын насихаттаулары тиіс. Кез келген ғылыми әдебиет визуалды тілдердің көптеген тағайындауын көрсетеді:

Программалаудың визуалды тілдерді түсінудегі жеңілдік (Ease of Comprehension) визуалды сипаттау программаларды жеңіл түсінуге көмектесуі тиіс. Жекелеген графикалық елестетулерді де, олардың қысқалығы, дәлдігін және т.б. бағалауға болады. Бұған қарап, визуалды синтаксис тәжірибелі программалаушыларға қарағанда, техникалық программалаушы емес адам үшін көбірек пайдалы болатындығын ескеру қажет.

Парадигма қолдауы (Paradigm Support) тәсілдердің артықшылығын немесе керісінше *программалау немесе парадигмалардың* кез келген әдісін білдіреді. Программалық қамтамасыздандыруды өңдеуде маңызды әсер ететін мұндай парадигмалар жұмыс барысында қажет. Программалау тілі нағыз тиімді тіл болу үшін таңдалған парадигманы қолдауы тиіс. Визуалды программалау бойынша бақылау-бағалау материалы түрлі парадигмалардың синтаксисі мен семантикасы мәселелеріне қатысты тапсырмаларды қамтуы тиіс. Объектіге-бағытталған программалауға ерекше назар аудару керек.

Визуалды программалау тілі өзінің негізгі парадигмасына қолдау көрсетуі немесе көптеген парадигмаларға жақын болуы тиіс деп ойлаймыз (мысалы, Vista визуалды тілі, мульти-парадигмаларды қолдаумен қамтамасыз етеді. Біздің критерий нақты визуалды елестетуге арнайы бейімделген программалаудың жаңа парадигмаларының пайда болу мүмкіндігін жоққа шығармайды.

Масштабтылық (Scalability) соңғы 25 жылда программалық қамтамасыз етуді өңдеудің әдістері, әдістемесі және құралдарының масштабтылығының болмауынан туындап отыр. Аз және орта программалық артефактарда жақсы жұмыс істейтін әдістер мен құралдар одан да үлкен артефактарда жұмыс істеуі міндетті емес. Бұл үшін болашақ информатика мамандары мәтіндік тілдерде аталмыш мәселені шешудің әдістері мен көмекші тәсілдерін білуілері қажет. Аналогиялық мәселелер программалаудың визуалды тілдеріне әсер етеді. Егер визуалды программалаудың жекелеген тілі шағын программаларда ғана қолданылса, онда оны толық тіл деп есептеуге болмайды. Визуалды тілдердің бұл ерекшелігі оларды «төмен деңгейлі» тілдерге жатқызуға негіз болады.

«Мұндай тілдердің программаға қажетті барлық тым кіші логикаға дейін, соның ішінде шартты және қайталауларды да (рекурстар немесе итерациялар) жеткізе алатын барлық мүмкіндіктері болады, алайда программа бөлімдерін модульдерге жіктеуге болмайды».

Масштабты қамтамасыз ету үшін модульдікті синтаксистік қолдау, ақпаратты абстракциялау және жасыру қажет.

Программалаудың визуалды тілдерін бағалау критерийлері мен визуалды программалауға оқытудың мазмұны негізінде біздің тарапымыздан болашақ информатика мамандарының жадыларында ЖОО бітірген соң сақталып қалуы тиіс оқытудың келесідей нәтижелері анықталды:

Оқытудың нәтижелерін құзіреттіліктер арқылы сипаттаймыз және әрі қарай қолданысқа ыңғайлы болуы үшін код тағайындаймыз:

C_1 – оқиғалы объектіге бағытталған тілдер туралы түсінік, визуалды архитектура.

C_2 – Borland C++ Builder зерттеменің интеграцияланған ортасы, оның элементтері.

C_3 – C++тілі және оның элементтері.

C_4 – C++ қосымшасын жобалау.

C_5 – C++ қосымшасын кодтау.

C_6 – C++ қосымшасында DLL кітапханасы көмегімен қосымшасын кодтау.

C_7 – Active X құрылыс клиенттері, Active X серверлері, Active X басқару элементтері, Құжаттамалар, Active X меңгеру элементтері, құжаттардың бірнеше интерфейстері.

C_8 – графикалық программалау.

C_9 – мәліметсіз программалау.

Сонымен біздің тарапымыздан визуалды программалау саласында кәсіби құзіреттіліктің 9 құзіреттілігі ажыратылды. Бұл жағдайда болашақ информатика мамандарының немесе оқытушының таңдауы бойынша Си класының орнына программалаудың басқа тілі таңдалып алынуы мүмкін.

Визуалды программалау саласында болашақ информатика мамандарында құзіреттіліктің қалыптасу деңгейін анықтау үшін құзіреттіліктің қалыптасу деңгейін атап көрсету қажет. Программалау үдерісі күрделі болғандықтан, құзіреттіліктің қалыптасу деңгейін анықтау үшін Блум таксономиясын қолданамыз:

1. Білу (К) – визуалды программалаудың базалық ұғымдарын, терминдерін, визуалды программалаудың ережелері мен қағидаларын меңгерген білім алушы C++ - q_1 белгісі.

2. Ұғыну, түсіну (С) – білім алушы типтік тасырмаларға жоспарлау тұжырамдамаларын қолдана алады, құбылыстарды модельдей алады, жобалау қосымшасының кемшіліктерін көрсете алады, стандартты ережелер бойынша алгоритмдерді және қарапайым программаларды жүзеге асыра алады; q_2 – берілген шаблон бойынша алгоритмді (программаны) қайта құрастыра алады.

3. Қолдану (AP) – қосымша жобалаудың берілген шарттары мен жаңа жағдайларында белгілі алгоритмдерді, программалаудың ережелері мен әдістерін қолдана алу, жаңа жағдайлардағы формалармен жұмыс жасаудың меңгерілген алгоритмдерді қолдана білу, q_3 – нақты тәжірибелік жағдайларда әдістерді қолдана алу.

4. Талдау (AN) – тапсырмаларды ішкі тапсырмаларға дұрыс бөле алу, объектінің, үдерістің құрылымын анықтау, ақпараттық жүйеде және ғылыми зерттеу объектісінде бүтінді бөлшекке бөле білу, оладың арасындағы байланысты атап көрсету, бүтінді ұйымдастыру қағидаларын тани алу қабілеті; q_4- визуалды программалау тәсілдер тапсырмаларын шешуге арналған мәліметтерді бағалаудың ақпараттық тапсырмаларын шешу логикасында айқын емес қателіктер мен жетіспеушіліктерді атап көрсете алу.

5. Жинақтау (S) - q_5 – визуалды программалауда туындайтын стандартты емес, проблемалық жағдайларда шарттарды өзгерте білу, құрастыра алу, жобалай және жоспарлай алу.

6. Бағалау (E) – жаңа мәліметтермен қызықтыратын жобаны, өз тәжірибесін және сарапшылардың тәжірибесін жүзеге асырудың нақты және жоспарлы жолын салыстыру негізінде ақпараттық тапсырмаларды шешудің модельдерін атап көрсете алу және түзете алу, жоба мақсатын өзгерту деңгейінде стратегиялық тұрғыдан ойлай білу, q_6 – ақпараттық жүйелердің мінез-құлығын басқару.

Іс-әрекет түрлері бойынша критерийлерді құрастыру

Іс-әрекеттің құрамдас бөліктері	**Бағалау критерийлері**
Есепті талдау (C_1)	• кіріс және шығыс объектілері мен олардың компоненттері арасындағы негізгі қажетті қатынастар - q_1; • мәселені шешу үшін орындалуы керек ішкі мақсаттар - q_2; • алгоритмнің спецификациясы - q_3.
Алгоритмін құру (C2)	• Стандартты шаблондар кітапханасын қолдану (Standard Template Library) C++ - q_4; • негізгі алгоритмдерді өз бетінше жүзеге асыру - q_5 • жаңа есептеу орталарын қолдану (аппараттық және бағдарламалық) - q_6; • құралдарды жүзеге асыру: Компоненттік оқиғалар тосқауылдарын (перехватчиков) құру - q_7; • құралдарды жүзеге асыру: Аралық кодтарты генерациялау шеберлері - q_8; • әрекет ағашы арқылы бағдарламалық кодты визуализациялау, ғылыми жаңалық элементтерінің болуы - q_9

Алгоритмді негіздеу (С3)	• кез келген көлемдегі жобалармен жұмыс істеу - q_{10}; • программаны түзету құралдарының болуы - q_{11}; • «алгоритмді құру бойынша» негіздемесі - q_{12}; • қадамдарды дұрыс детальдау - q_{13} • жеке дәлелдемелер - q_{14}
Сынақ (тест) топтамасын таңдау және негіздеу (С4)	• сынақ (тест) топтамасының толықтығы - q_{15}; • барлық нәтижелердің сипаттамасының дұрыстығы - q_{16}; • тармақталуы - q_{17};
Бағдарламаның сапасы (С5)	• программадағы басқарудың құрылымдалуы - q_{18} • Түсініктемелер мен мағыналы (мнемоникалық) атауларды қолдану - q_{19} • программадағы жолдардың ұзындығы - q_{20}; • қайта пайдалануға болатын фрагменттерді процедуралау (қайта өңдеу) - q_{21} • көмекші айнымалыны қолдану - q_{22};
Бағдарламаны орындау үшін рәсімдеу (C_6)	• нәтижелерді визуализациялау - q_{23}.

Қажеттіліктің жалпы қызмет түрін анықтайтын шарттарды тексеру

Құзіреттілік коды	Құзіреттіліктің қалыптасу бағасы	Жеке бағалар
С1	$q_1 \times 0{,}05 + q_2 \times 0{,}05 + q_3 \times 0{,}1$	m_1
С2	$q_4 \times 0{,}2 + q_5 \times 0{,}1 + q_6 \times 0{,}1 + q_7 \times 0{,}1 + q_8 \times 0{,}1 + q_9 \times 0{,}2$	m_2
С3	$q_{10} \times 0{,}1 + q_{11} \times 0{,}15 + q_{12} \times 0{,}25 + q_{13} \times 0{,}2 + q_{14} \times 0{,}3$	m_3
С4	$q_{15} \times 0{,}2 + q_{16} \times 0{,}15 + q_{17} \times 0{,}2 + q_{18} \times 0{,}45$	m_4
С5	$q_{19} \times 0{,}65 + q_{20} \times 0{,}35$	m_5
С6	$q_{21} \times 0{,}35 + q_{22} \times 0{,}25 + q_{23} \times 0{,}4$	m_6

Критерийлер мағынасын біріктіру қажеттіліктің мульти-критериалды функциясы (альтернативтердің критерийлері және жалпы сапасы бойынша альтернатив бағаларының арасындағы тәуелділікті анықтау):

$$a1 = 0{,}15 \times m_1 + 0{,}15 \times m_2 + 0{,}15 \times m_3 + 0{,}15 \times m_4 + 0{,}15 \times m_5 + 0{,}4 \times m_6$$

ҚОРЫТЫНДЫ

Программалау бойынша оқу жетістіктерін бағалаудың мазмұны программалаудың пәндері бойынша жұмыс бағдарламалары негізінде келесідей жалпы элементтер арқылы сипатталды:

- программалауды білуді бағалау: программалау тілінің синтаксисі (программалаудың визуалды тілі ретінде C++, C#, және Visual Studio орталары қолданылады), программалау мен концепциялар қағидалары, программалау саласында білімді қолдану, кодты түсіну;

- программалау дағдыларының қалыптасуын бағалау:мәселелерді шешу мен жоспарлау, программаларды ретке келтіру, түсінікті кодты жазу (кодты жүзеге асыру), шешімдерді құру.

Программалау бойынша оқу жетістіктерін бағалаудың оқу қызметі түрлері бойынша технологиясы бөлімінде бір критериалды модельдер негізінде біліктілік сипаттамасының жеке бағаларының қорытындысын, жеке бағалардың болжамдық талдауын, барлық жеке бағалардың мағыналары қатаң түрде бір альтернативаның интервалына түсіп тексеріледі, жеке бағалардың мағыналары көрші альтернативаның интервалына түсіп (дәреже есебімен) тексеріледі, кешенді бағалаудың критериалды қорытындысын, кәсіби құзіреттіліктің қалыптасу деңгейінің қорытынды бағалары анықталды.

Бұл технологиялар білім алушылардың программалау бойынша оқу жетістіктерін объективті және дәл бағалауға мүмкіндік береді. Бағалау процесінде критериалды тәсілді қолдану арқылы, студенттердің алған білімдері мен дағдылары нақты критерийлер бойынша салыстырылады.

Программалау пәндерінде қолданылатын бағалау құралдарына тестілеу, практикалық жұмыстар, жобалар, курстық жұмыстар және емтихандар кіреді. Тестілеу білім алушылардың теориялық білімдерін тексеруге бағытталған болса, практикалық жұмыстар олардың программалау дағдыларын және мәселелерді шешу қабілеттерін бағалайды. Жобалық жұмыстар мен курстық жұмыстар студенттердің шығармашылық қабілеттерін және күрделі мәселелерді шешу мүмкіндігін көрсетеді.

Сонымен қатар, оқу процесінде формативті және суммативті бағалау түрлері қолданылады. Формативті бағалау білім алушылардың оқу барысындағы жетістіктерін қадағалауға және оларды уақытында түзетуге мүмкіндік береді. Суммативті бағалау болса, оқу кезеңінің соңында жалпы жетістіктерді қорытындылауға арналған.

Бағалау нәтижелерін талдау барысында, білім алушылардың қандай салаларда қиындықтарға тап болатындығын анықтауға болады. Бұл ақпарат оқытушыларға оқу бағдарламаларын жетілдіруге және білім алушылардың қажеттіліктеріне сәйкес түзетулер енгізуге мүмкіндік береді. Кәсіби құзыреттіліктің қалыптасу деңгейін анықтау үшін кешенді бағалау қолданылады. Бұл бағалау білім алушылардың теориялық білімдерін, практикалық дағдыларын және кәсіби қасиеттерін біртұтас түрде қарастырады.

Нәтижесінде, оқу процесінің тиімділігі артады және білім алушылар болашақта еңбек нарығында сұранысқа ие маман ретінде қалыптасады.

Жалпы, программалау бойынша оқу жетістіктерін бағалау технологиялары білім алушылардың білім сапасын арттыруға, олардың кәсіби құзыреттіліктерін дамытуға және оқу процесін оңтайландыруға ықпал етеді. Болашақта бұл технологияларды әрі қарай жетілдіру және жаңа әдістерді енгізу арқылы білім беру саласында жоғары нәтижелерге қол жеткізуге болады.

Сонымен қатар, заманауи цифрлық құралдарды қолдану бағалау процесін жеңілдетеді және нәтижелердің дәлдігін арттырады. Мысалы, онлайн платформадағы автоматтандырылған тестілеу жүйелері студенттердің білім деңгейін жылдам және тиімді бағалауға мүмкіндік береді.

Оқытушылардың кәсіби біліктілігін арттыру да маңызды фактор болып табылады. Олар жаңа бағалау әдістерін, технологияларын меңгеріп, оларды практикада қолдана білуі тиіс. Бұл білім алушылардың оқу мотивациясын көтеріп, олардың пәнге деген қызығушылығын арттырады.

Сондықтан, программалау бойынша оқу жетістіктерін бағалаудың мазмұны мен технологиясын үнемі жаңартып отыру қажет. Бұл білім беру сапасын жоғарылатып, қоғамның қажеттіліктеріне сәйкес білікті мамандарды дайындауға септігін тигізеді.

Әдебиеттер

1 Огнёва М.В., Кудрина Е.В. Программирование на языке С++: практический курс. Учебное пособие для СПО, Litres, 2017

2 Касьянов В.Н., Сабельфельд В.К. Сборник заданий по практикуму на ЭВМ: учебное пособие для вузов. – М.: Наука, 1986. – 272 с.

3 Поляков А.Ю., Брусенцев В.А. Методы и алгоритмы компьютерной графики в примерах на Visual C++. – Изд. 2-е, перераб. и доп. – СПб.: БХВ-Петербург, 2013. – 560 с.

4 Панюкова Т.А., Панюков А.В. Языки и методы программирования. Создание простых GUI-приложений с помощьюVisual C++: учебное пособие. –М.: Мир, 2015. – 144 с.

5 Лаптев В.В., Толасова В.В. Модели оценивания в обучающей системе по программированию // Вестник АГТУ. Серия: управление, вычислительная техника и информатика. – 2009. – №1. – С. 187-192.

6 Brooks F.P. No silver bullet-Essense and accident in software Engineering // IEEE Computer. – 1987. – Vol. 20. – P. 10-19.

7 Lohse G.L., Biolsi K., Walker N., Rueter H.H. A classification of visual representations // Communications of the ACM. – 1994. – Vol. 37. – P. 36-49.

8 Burnett M.M., Goldberg A., Lewis T.G. eds. Seven Programming Language Issues // In book: Visual Object-Oriented Programming: concepts and environments. – Greenwich, CT: Manning Publishing, 1995. - P. 161-182.

9 Heydon A., Maimone M.W., Tygar J.D., Wing J.M., Zaremski A.M. Miro: visual specification of security // IEEE Transactions on Software Engineering. - 1990. - Vol. 16. – P. 1185-1197.

10 Graf M. Building a visual designer’s environment // In book: Principles of Visual Programming Systems / Ed. Shi-Kuo Chang. – Englewood Cliffs, NJ: Prentice-Hall, 1990. – P. 291-325.

11 Wilde N., Lewis C. Spreadsheet-based interactive graphics: from prototype to tool // Conf. on Human Factors in Computing Systems. – Seattle; WA, 1990. – P. 153-159.

12 Ames C., Kiper J., Howard E. Using program visualization to enhance maintainability and promote reuse // 10th Computing in Aerospace conf. / American Institute of Aeronautics and Astronautics. – San Antonio, 1995. – P. 540-551

13 Ambler A.L., Burnett M.M., Zimmerman B.A. Operational versus definitional: a perspective on programming paradigms // IEEE Computer. – 1992. – Vol. 25. – P. 28-43.

14 Schiffer S., Frohlich J. H. Visual programming and software engineering with Vista // In book: Visual Object-Oriented Programming / Eds.M. Burnett, A. Goldberg, T. Lewis. - Greenwich, CT: Manning Publication, 1994. – P. 199-227.

15 De Remer F., Kron H.H. Programming in the large versus programming in the small // IEEE Transaction on Software Engineering. – 1976. – Vol. SE-2. – P. 80-86.

16 Basili V., Rombach H. The MTAMEN project: towards improvement-oriented software environments // IEEE Transactions on Software Engineering. – 1988. – Vol. 14. – P. 24-33.

17 Bloom B.S. ed. Taxonomy of educational objectives: The classification of educational goals: handbook I, cognitive domain. – London: Longmans, 1956. – 207 p.

МАЗМҰНЫ

Ғылыми басылым

ПРОГРАММАЛАУ НЕГІЗДЕРІ БОЙЫНША ЕСЕПТЕРДІ ШЕШУ ПРАКТИКУМЫ

оқу-әдістемелік құрал

Токжигитова Н. К.

Программалау негіздері бойынша есептерді шешу практикумы: [Мәтін:] оқу-әдістемелік құрал. Құраст.: Н. К. Токжигитова. «Toraighyrov University» КеАҚ. Семей: «Zhardem» республикалық баспа компаниясы. – 2023 жыл. – 100 бет.

Материалдың дұрыс болуына, грамматикалық және орфографиялық қателерге авторлар мен құрастырушылар жауапты

Техникалық редактор Смағұлова А.Т.
Шығарушы директор А.Ғали-Арыстанұлы
Кезекші редактор А.Тойшыбекова
Беттеуші дизайнер: Какитаев А.А.
Техникалық редактор А.Ержанқызы
Корректор Ж.Құдайбергенова

Терімге жіберілді 25.01.2023. Басуға қол қойылды 27.12.2023.
Қалпы 60×84/8 Көлемі 6,25 б.т. Қағазы офсетті. Қаріп түрі әдеби.Баспа табағы 6,25. Тираж 300 экз. Тапсырыс № 02221. Бағасы келісім негізінде

«Zhardem» республикалық баспа компаниясы
071710, Семей қаласы., Гоголь көшесі, 89 үй
Тел./факс: 8 7222 52-13-72

Тапсырыс берушінің файлдарынан
Қазақстан Республикасы
«Zhardem» баспа компаниясының баспаханасында басылды.
071710, Семей қаласы, Гоголь көшесі, 89
Zhardem.baspasy@bk.ru

ISBN 978-601-345-503-7

978-601-345-503-7