ердің тірек
конспектісі

Нысан
ПМУ ҰСН 7.18.2/05

Қазақстан Республикасының Білім және ғылым министрлігі

С. Торайғыров атындағы Павлодар мемлекеттік университеті

Информатика және ақпараттық жүйелер кафедрасы

5В070300 «Ақпараттық жүйелер» мамандығының студенттеріне
«Алгоритмдер, деректер құрылымдары және бағдарламалау» пәнінен

ДӘРІСТЕРДІҢ ТІРЕК КОНСПЕКТІСІ

Павлодар

1 Кіріспе

Тілдің алфавитінде латын бас және кіші әріптері, цифрлар және арнайы символдар (нүкте, үтір, апостроф, қос нүкте) пайдаланылады.

Ескертпелер (комментария) /* және */ символдары арасында жазылады. Немесе // символдың артында жазылады.

Тілдегі айнымалы, функция, объект аты ретінде пайдаланылатын идентификатор ұғымы негізгі болып табылады. Идентификатор 32 символдан тұруы мүмкін. Оның аты әріптерден басталу қажет. Онда әріптер мен цифрлар пайдаланылады. Си тілі регистрге сезімтал. Мысалы: SUMM, Summ, summ айнымалылары әр – түрлі болып есептеледі.

СИ тілінде мәліметтердің келесі типтері бар.

1. **int** (бүтін) –32768 -ден 32767-ға дейінгі аралықта мәндерді қабылдайтын бүтін сандардың типі. int типті айнымалы жадыда 16 бит орын алады;
2. **short** (қысқа бүтін) – осы типтегі айнымалылар int типті айнымалылардан артық болмау керек. Осы типтегі айнымалы жадыда 16 бит орын алады;
3. **long** (ұзын бүтін) – –2147483648 –дан 2147483647-ға дейінгі аралықта мәндерді қабылдайтын бүтін тип. long типті айнымалы жадыда 32 бит орын алады;
4. **char** (символдық) – мәндері символ болатын тип;
5. **unsigned** (белгісі жоқ) – СИ тілінде кейбір (char, short, int, long) типтерді unsigned модификаторы көмегімен тек қана оң сандарды қабылдауға мүмкіндік беретін тип. int типін сипаттағанда «unsigned int a;» орнына «unsigned a;» деп жазуға болады;

Мәліметтер типі	Min	Max
255signed char	- 128	127
signed short0 unsigned char	-32768	32767
unsigned short	0	65535
signed int	-32768	32767
unsigned int	0	65535
signed long	-2147483648	2147483647
unsigned long	0	4294967295

6. **float** (нақты) – мәндері нақты сан болатын тип. Нақты сандар экспонента түрінде жазылуы мүмкін, мысалы, -1.58e+2 (ол $-1{,}58 \cdot 10^2$ тең). float типті айнымалы 32 бит орын алады. Оның мәні +3.4e-38 –ден +3.4e+38-ға дейінгі аралықта жатуы мүмкін;
7. **double** (дәлділік) – float типті айнымалыдан қарағанда жадыда екі есе орын алатын (64 бит) нақты айнымалыларды анықтайды. Олардың мәндері +1.7e-308-дан +1.7e+308-ға дейін болуы мүмкін.
8. **long double** (ұзын нақты) – типті айнымалы жадыда 10 байт орын алады. Олардың мәндері 3.4E-4932 – дан 3.4E+4932- ға дейін болуы мүмкін.

Пайдаланылатын әдебиеттер [1, 2]

2 СИ++ бағдарламалау тілінің негізгі элементтері

Шығару операторы

Жазылуы: printf("басқару жолы", аргумент1, аргумент2, ...);

Басқару жолында үш типті объект жазылуы мүмкін: қарапайым символдар, түрлендіру спецификациясы, басқарушы символ-константалар.

Әр- бір түрлендіру спецификациясы % белгісінен басталып, түрлендіру жасайтын символмен аяқталады. Түрлендіру символы айнымалы типіне байланысты болады:

1) d – аргумент мәні ондық бүтін сан;
2) o – аргумент мәні сегіздік бүтін сан;
3) x – аргумент мәні он алтылық бүтін сан;
4) c – аргумент мәні символ;
5) s – аргумент мәні символдар жолы;
6) e – аргумент мәні экспонента түріндегі нақты сан;
7) f – аргумент мәні жылжымалы нүктемен нақты ондық сан;
8) u – аргумент мәні белгісі жоқ бүтін сан;
9) p – аргумент мәні көрсеткіш (адрес).

Егер % белгісінен кейін символ жазылмаса, онда экранға сол белгінің өзі шығады. printf функциясы қанша аргумент бар және олардың типтері қандай екенін анықтау үшін басқару жолын пайдаланады.

Мысалы, %6f – алты позициядан тұратын өрісте жылжымалы нүктемен санды шығару; %.2f – нүктеден кейін екі цифрмен нақты санды шығару; %6.2f – нүктеден кейін екі цифрмен нақты санды алты позициядан тұратын өріске шығару.

Мысалы, a=3,687 және b=10,17 болса, онда

printf(“%7f %8f”,a,b);

командасынан кейін экранда нәтиже келесідей шығарылады:

_ _ 3.687 _ _ _10.17

(7 поз.) (8 позиция)

Мысалда көріп отырғандай бос позициялар пробелмен толтырылады.

Егер printf(“%.2f %/2f”, a, b); функциясын пайдаланса онда нәтиже келесідей болады:

3.69 10.17 ,

Егер printf(“%7.2f e”,a,b); функциясын пайдаланса онда нәтиже келесідей болады:

_ _ _ 3.68 1.010000e+01

(7 позиция)

Спецификация алдындағы «минус» таңбасы санды сол жағынан шығарады. Мысалы:

printf(“%-6d”,s);

функциясынан кейін экранда келесі жазу шығады:

336_ _ _

(6 позиция)

Ескертетін жағдай, егер өріс ұзындығы сандағы цифрлардың санынан кем болса, онда өріс қажетті өлшемге дейін ұзарады.

Жоғарыда айта кеткен басқарушы символ-константалардың ішінен көп пайдаланылатын келесілер:

1) \a – қысқа дыбыс сигналын шығару үшін;
2) \b – меңзерді бір позицияға солға жылжыту;
3) \n – жаңа жолға көшу үшін;
4) \r – меңзерді жолдың басына қою үшін;
5) \t – көлденең табуляциялау;
6) \v – тігінен табуляциялау.
7) \0 – бос литера, NULL

Мысалы, i=50 болсын, келесі функцияның жазуынан кейін

printf(“\t ЭЕМ\n%d\n”,i);

алдымен көлденең табуляция (\t) орындалады, яғни экран шетінен 8 позиция орын қалдырады. Содан соң ЭЕМ сөзі шығып, меңзер келесі жолдың басына түседі (\n), d

форматы бойынша і-дің бүтін мәні шығады. Соңында меңзер қайтадан келесі жолға түседі. Сонымен экранда келесідей нәтиже шығады:

_ _ _ _ _ _ _ _ ЭЕМ

50

Пайдаланылатын әдебиеттер [1, 2]

3 Си тілінің операторлары

Си тілінде меншіктеу операторы «=» белгісімен белгіленеді. Жазылуы:

айнымалы = мән;

Басқа тілдерге қарағанда Си тілінде меншіктеу операторын өрнектерде пайдалануға болады. Мысалы, if ((c=a+b)<0) printf("c саны нөлден кіші').

Тағы бірнеше рет меншіктеу мүмкіндігі бар. Мысалы:

x=y=z=a*b;

Бұл жерде алдымен a*b мәні z-қа, содан соң y-қа, сондан кейін x-қа меншіктеледі.

Си тілінде константалардың мәнін өзгертпеу үшін const модификаторы пайдаланады. Мысалы: const float a=3.5;

const j=47;

Өрнектерді айнымалыларды есептеуге арналған формула ретінде пайдаланады. Олар *операндтардан* (айнымалылар, константалар және т.б.) құралады және операция белгілерімен (қосу, азайту, көбейту және басқамен) байланысады.

Кесте 1 – Арифметикалық операциялар

Арифметикалық операциялар	Орындалуы
+	Қосу
–	Азайту
*	Көбейту
/	Бөлу, бүтін бөлігін алу
%	Бөлгендегі қалдығын алу

Мысалы: егер b=5, c=2, бүтін типті айнымалылар болса, онда

a=b%c; d=b/c;

операциялардың орындалуынан кейін a айнымалысы 1 мәнін қабылдайды, d айнымалысы 2 мәнін қабылдайды.

Арифметикалық операциясына сәйкес меншіктеу операторын басқаша пайдалануға болады.

Мысалы, x=x+n; жазуын x+=n; деп жазуға болады. Сол секілді

x=x-n; x-=n;

x=x/n; x/=n;

x=x*n; x*=n;

x=x%n; x%=n;

Кесте 2 – Қатынас операциялары

Қатынас операциялары	Ұғымы
<	Кіші
<=	Кіші не тең
= =	Тең
>=	Үлкен не тең
>	Үлкен
!=	Тең емес

Кесте 3 – Логикалық операциялар

Логикалық операциялар	Ұғымы
&&	ЖӘНЕ, and
||	НЕМЕСЕ, or
!	ЕМЕС, not

Си тілінде логикалық тип жоқ болғандықтан, логикалық өрнектін нәтижесі сандық шама болады (жалған мәнде 0-ге тең, ақиқат мәнде 0-ге тең емес шама).

Си тілінде операнд мәнін бірге арттыру (++) және бірге кеміту (--) операциялары пайдаланады.

x++; ++x; операторлардың нәтижесі бір, бірақ пайдалану барысында айырмашылықтары бар. Мысалы:

```
main()
{
int x,y;
x=5; y=60;
x++;  ++y;
printf("x=%d  y=%d\n",x,y);
printf("x=%d  y=%d\n",x++,++y);
}
```

Нәтижесі:

x=5 y=61

x=5 y=62

x++ операторын пайдаланғанда x айнымалының мәні алдымен өрнекте пайдаланады, содан соң бірге артады. ++y пайдаланғанда алдымен y айнымалысы бірге артады, содан соң өрнекте пайдаланады.

Кесте 4 - Операциялардың орындалу приоритеті

++ --	* / %	+ -	!	< <= > >=	== !=	&&	||

Егер операция приоритетін өзгерту қажет болса, дөңгелек жақшаны пайдаланады. Олардың приоритеті жоғары.

Өрнектерде тағы шарт операциясы ? кеңінен пайдаланады.

y=x?a:b,

өрнегінде егер x нөлге тең болмаса, y=a, егер x нөлге тең болса y=b.

y=(a>b)?a:b;

өрнегі y айнымалысына a не b айнымалылардың ең үлкен мәнін қабылдауға мүмкіндік береді, яғни y=max(a,b).

Пайдаланылатын әдебиеттер [1, 24-56 бет; 3, 15-46 бет; 5]

4 Массивтер, құрылымдар және көрсеткіштер. Динамикалық массив

Массив – бұл элементтері бір типті болатын мәліметтер жиыны.

Массивтерді айнымалыларды секілді сипаттайды:

```
int a[100];
float c[10][20];
```

Біріншісі бүтін типті a[0],a[1], … ,a[99] (индекстелу нөлден басталады) элементтерден тұратын бір өлшемді массив болып табылады. Екіншісі элементтері нақты тип болатын екі өлшемді массив болып табылады. Ол 10 жолдан, 20 бағаннан тұрады.

Массивті келесідей инициализациялауға болады:

int a[6]={10,20,30,40,50,60}; немесе int a[]={10,20,30,40,50,60};

float[2][3]={{10,20,30},{40,50,60}};

Мысал 1. 10 нақты саннан тұратын бірөлшемді S массиві берілген. Осы массивті кері ретпен шығару.

```
#include <stdio.h>
main()
{float s[10];
int i;
for (i=0;i<10;i++)
  scanf(“%f”,&s[i]); /*массив элементтерін енгізу*/
for (i=9;i>=0;i--)
  printf(“%f”,s[i]); /* массив элементтерін кері ретпен шығару*/
}
```

Мысал 2. Бес бағаннан және бес жолдан тұратын бүтін типті екі өлшемді массив берілген. Массивтегі ең үлкен элементтің орналасу номерін анықтау.

```
#include <stdio.h>
main()
{int i,j,max,in,jn,a[5][5];
for (i=0;i<5;i++)
for (j=0;j<5;j++)
scanf(“%d”, &a[i][j]); /*матрицаны енгізу*/
max=a[1][1];
for (i=0;i<5;i++)
for (j=0;j<5;j++)
if (a[i][j]>max)
{in=i; jn=j;}
printf(“%d %d”, in, jn);
}
```

Мысал 3. D(6,6) массивінің басты және кері диагональдің орындарын ауыстыру.

```
#include <stdio.h>
main()
{int i,j,a,d[6][6];
for (i=0;i<6;i++)
for (j=0;j<6;j++)
scanf(“%d”, &d[i][j]);

for (i=0; i<5; i++)
{a=d[i][i];
d[i][i]=d[i][6-i];
d[i][6-i]=a;
}
for (i=0; i<6; i++)
for (j=0; j<6; j++)
printf(“%d%c”, d[i][j], (j==5)?”\n”:” “);
}
```

Матрица элементтерін жолдар бойынша шығаруда ? операциясы пайдаланылады. Егер j=5, онда курсор келесі жолға түседі (“\n”), басқа жағдайда пробел шығарылады. Экранға символдық шама шығарылатындықтан %c спецификациясы пайдаланылады.

Пайдаланылатын әдебиеттер [1, 56-102 бет; 3, 46-87 бет]

5 Функциялар және ішкі программалар

Басқа жоғары деңгейдегі бағдарламалау тілдеріне қарағанда Си тілінде ішкі программалар функция мен процедураға бөлінбейді. Бағдарлама тек функциядан құрылады. Функция – бұл нақты есепті шешуге арналған операторлардың және өрнектердің жиыны. Функция арасындағы байланыс аргументтер, нәтиже мәндері және сыртқы айнымалылар арқылы жасалады.

Мысалы:

```
float srzn(int a, int b)
{
int y;
y=(a+b)/2;
return(y);
}
```

Функция нәтиже ретінде бір мәнді бере алады. Ол үшін қайтару операторы пайдаланылады:

return (өрнек);

Бұл жердегі өрнек айнымалы болуы мүмкін.

Мысал 1: санның абсолютті шамасын функция көмегімен есептеу.

```
#include <stdio.h>
#include "abc.cpp"
main()
{
int a=10, b=0, c=-20;
int d,e,f;
d=abc(a);
e=abc(b);
f=abc(c);
printf("%d %d %d", d, e, f);
}
```

Функцияның өзі

```
abc(x)
int x;
{int y;
y=(x<0)?-x:x;
return(y);
}
```

Көрсетілген бағдарламада функция типі сипатталмаған. Ол тек қана функция нәтижесі бүтін типті болғанда мүмкін. Басқа жағдайда типті сипаттау қажет.

Және де осы бағдарламада функция бөлек файлда abc.cpp сақталған және #include процедурасымен қосылған. Тырнақша (“ “) белгісімен қоршалған файл ағымды каталогтан ізделінеді (бағдарлама мен функция ағымды каталогта сақталу керек), ал < > жақшаға алынған файл кітапхана катологында ізделінеді.

Функцияны бағдарламаның ішінде де сипаттауға болады.

Мысал 2: f=$\sqrt{x} + y / z$ формуласын функция көмегімен есептеу.

```
#include <stdio.h>
double vv(x,y,z)
double x, y, z;
{
```

```
double f;
f=sqrt(x)+y/z;
return(f);
}
main()
{
double x=5.5, f, y=10, z=20.5;
f=vv(x,y,z);
printf("%f ", f);
}
```

Егер функция main() функциясынан кейін сипатталса оның прототипін көрсету қажет.

Мысалы: алдыңғы есеп

```
#include <stdio.h>
double vv(double x, double y, double z);
main()
{.......}
double vv(x,y,z)
double x, y, z;
{......}
```

Си тілінде сонымен қатар, аргументі жоқ функцияны және ешқандай нәтиже қайтармайтын функцияны пайдалануға болады. Нәтиже қайтармайтын функцияға void типін пайдалану қажет. Ол қайтарылатын мәннің болмауын білдіреді.

Мысал 3: a және b айнымалылардың орнын ауыстыру.

```
#include <stdio.h>
#include "izm.cpp"
main()
{ int a,b;
scanf("%d %d", &a, &b);
izm(&a,&b);
printf("%d %d", a, b);
}
```

izm.cpp файлында сақталған функция

```
void izm(a, b);
int *a, *b;
{ int c;
c=*a;
*a=*b;
*b=c;
}
```

Егер функция аргументі ретінде массив аты пайдаланса, онда оған массивтің басталу адресі беріледі. Элементтердің өздері көшірілмейді. Функция массив элементтерін индекстеу арқылы басынан жылжыта алады.

Мысал 4: S массивінің элементтерін орындарымен ауыстыру: біріншісін екіншісімен, үшіншісін төртіншісімен және т.с.

```
#include <stdio.h>
void reverse(s)
int s[];
{
int a,i;
for (i=1; i<5; i+=2)
{a=s[i]; s[i]=s[i+1]; s[i+1]=a;}
```

```
}
main()
{int i,j,s[6];
for (i=0; i<6; i++)
scanf(“%d”,&s[i]);
reverse(s); /* reverse функциясын шақыру*/
for (i=0; i<6; i++)
printf(“%d”,s[i]);
}
```

Бірөлшемді массивті int s[]; деп сипаттауға болады, ал екі өлшемді массивті сипаттағанда екінші жақшада баған саны жазылуы қажет, мысалы: a[][3].

Мысал 5. a(5,5) массивінің барлық элементтерін екіге арттыру.

```
#include <stdio.h>
void mas(a)
int a[][5]; /* a массивін сипаттау */
{int i,j; /* i,j айнымалыларын сипаттау*/
for (i=0; i<5; i++)
for (j=0; j<5; j++)
a[i][j] = 2*a[i][j]; /*массив элементтерін екі есе арттыру*/
}
main()
{int a[5][5];
int i,j;
for (i=0;i<5;i++)
for (j=0; j<5; j++)
scanf(“%d”,a[i][j]); /*массивті енгізу*/
mas(a); /* mas функциясын шақыру*/
for (i=0; i<5; i++)
for (j=0; j<5; j++)
printf(“%d”, a[i][j]); /*нәтижені экранға шығару*/
}
```

Пайдаланылатын әдебиеттер [1, 103-124 бет; 3, 87-105 бет; 5]

6 Препроцессорлық құралдар

include, # define, # undef директивалары

Си трансляторының препроцессор деп аталатын ендірілген құралы бар. Ол бағдарламаны компиляцияға дейін қарастырады (осыдан препроцессор термині шыққан) және бағдарламадағы барлық символдық аббревиатураларды сәйкес директиваларға ауыстырады, көрсетілген файлдарды қосады. Препроцессор үшін бағдарлама жолдары # символынан басталады.

Файлдарды қосу үшін # include директивасын пайдаланған болатынбыз.

Препроцессордың басқа директиваларын қарастырайық:

define <идентификатор> < ауыстыру>

Бұл директива бағдарламадағы идентификатор орнына ауыстыру мәтінін қояды. Егер директива келесі түрде болса:

define идентификатор (идентификатор ,…идентификатор)

онда ол аргументтері бар макроауыстырудың анықтамасы.

Мысал 1:

```
# define FOOR TWO*TWO
# define PX  printf(“x тең %d\n”,x)
#define FNT  “x тең % d\n”
# include<stdio.h>
# define TWO 2
main ( )
{    int x=TWO;
       PX;
       x=FOOR;
       printf(FNT,X);
}
```

Бағдарлама орындалған соң экранға келесі жазу шығады:

X тең 2

X тең 4

Мысал 2. Аргументтері бар # define директивасының мысалын қарастырайық. Ондай директивалармен өте мұқият жұмыс істеу керек.

```
# include<stdio.h>
# define KV(X) X*X
# define PR(X) printf(“X тең %d\n”,X)
main ( )
{     int x=4;
int z;
z=KV(x);
PR(z);
PR(x);
PR(KV(x+2));
pr(100/KV(2));
}
```

Бағдарламадағы KV(x) x*x ауыстырылады.

Бағдарлама жұмысының нәтижесі:

z тең 16

x тең 4

KV (x+2) тең 14

100/KV (2) тең 100

Бірінші екі жол түсінікті, ал KV (x+2) 36-ға, 100/KV (2) 50-ге тең болу керек, бірақ нәтижелері мүлдем басқа. Оның себебі X*X орнына X+2*X+2 және 100/2*2 қойылады. Егер KV (x+2) дұрыс шығарылсын десеңіз, формуланы келесідей жазу қажет:

```
# define  KV(X)  (X)*(X)
```

Онда KV (X+2) орнына (X+2)*(X+2) қойылады.

#undef ең соңғы анықталған идентификаторды жояды. Жазылу түрі:

#undef идентификатор

Мұндағы идентификатор – #define директивасымен анықталған идентификатор:

```
#undef  ESCAPE
```

Бұл процедураның орындалуынан кейін ESCAPE идентификаторы анықталмайды.

Пайдаланылатын әдебиеттер [2, 154 бет; 4, 147 бет]

7 C++ тілінде файлдармен жұмыс

Файлға тізбектей қол жеткізгенде ақпараттың алмастырылуы енгізу – шығару жүйесімен берілетін арнай буфер арқылы жасалады. Си тілінің компиляциясы енгізу –

шығаруды тізбектей келетін файлдар ағымы ретінде қарастырады. Әр ағым файлмен байланыстырылады. Файл мен ағым арасындағы байланыс оның ашылуы кезінде жасалады. Файлдың ашылуы fopen функциясымен жасалады. Бұл функция файлға көрсеткіш қайтарады.

Файлға көрсеткішті келесідей сипаттайды:

FILE * lst;

Мұндағы FILE- тип аты, ол stdio.h стандартты анықтамада сипатталған;

lst- файлға көрсеткіш (логикалық аты).

fopen функциясы бағдарламада келесідей шақырылады:

lst= fopen (файлдың физикалық аты, файлды пайдалану түрі);

Файлдың физикалық аты - сақталынған орнын көрсетеді, мысалы "D:zni.f”- D: дискісіндегі zni.f файлы үшін.

Файлды пайдалану түрі “w” (егер файлға мәліметтерді жазу керек болса), “r” (файлдан мәліметтерді оқу қажет болса) және “a” (файлдағы мәліметтерге тағы мәліметтерді қосу қажет болса) болуы мүмкін. Егер ондай файл болмаса, ол жасалады.

Файлға жазу үшін fprintf, fputs, файлдан оқу үшін fscanf, fgets кітапханалық функциялар пайдаланылады. Файлмен жұмысты аяқтағанда ол жабылу тиіс, мысалы:

fclose (lst)

lst- файлға көрсеткіш;

Мысал 1: Енгізілген мәтінді файлға жазу

```
#include <stdio.h>
main()
{ char s[50];
FILE *fl;
fl=open (“fayl.txt”, “w”);
scanf(“%s”, &s);
fprintf(fl, “%s”, s);
fclose(fl);
getchar();
}
```

Мысал 2: Файлдан жолдарды оқып экранға шығару

```
#include <stdio.h>
main()
{ char s[50];
FILE *fl;
fl=open (“fayl.txt”, “r”);
while(!feof(fl))
{
fscanf (fl, “%s”, &s);
printf(“%s”, s);
}
fclose(fl);
getchar();
}
```

Пайдаланылатын әдебиеттер [1, 103-124 бет; 3, 87-105 бет; 5]

8 Символдық жолдармен жұмыс

СИ тілінде символдар жолын енгізуге және шығаруға арналған екі ыңғайлы puts және gets функциялары бар. Оларды пайдалану мысалын қарастырайық:

```
#include<stdio.h>
main()
```

```
{
char q[40]; /*символдар жолын сипаттау*/
puts(“Символдар жолын енгізіңіз ”);
gets(q); /*символдар жолын енгізу */
puts(q); /* символдар жолын шығару */
}
```

Программаның жұмысы нәтижесінде экранда алдымен келесі мәтін пайда болады:

Символдар жолын енгізіңіз

Содан кейін кез келген символдық жолды енгізу қажет. Енгізілген жолдың әрбір символы gets операторының көмегімен q массивінің элементтеріне меншіктеледі. Символдарды puts операторы шығарады.

Пайдаланылатын әдебиеттер [1, 103-124 бет; 3, 87-105 бет; 5]

Пайдаланылатын әдебиеттің тізімі

1. Подбельский В.В. Язык Cu ++: Учебное пособие. - М.: Финансы и статистика,2005, - 560 с.
2. Страуструп Б. Язык программирования Сг ++. - М.: Радио и связь, 2001. - 352 стр.
3. Собоцинский В.В. Практический курс Turbo Cu ++. Основы объктно- ориентированного программирования. - М.: Свет, 2003. - 236 с.
4. Романов В.Ю. Программирование на языке Cu ++. Практический подход. - М.: Компьтер, 2003. - 160 с.
5. Нургазина Б.К., Бельгибаева С.А. Алгоритмдеу және програм-малау тілдері: ақпараттық жүйелер мамандығы студент-теріне арналған оқу-әдістемелік құрал. Павлодар: Кереку, 2009
6. Юлин В.А., Булатова И.Р. Приглашение к Cu. - Мн.: Высш. Шк., 2000,- 224 с.