Тандоочу, токтоочу жана улантуучу операторлор Тандоочу оператор.

Көптөгөн маселелерде кандайдыр бир туюнтманын маанилерине карата бир канчалаган багыттар боюнча иш аракеттер аткарылат. Ошондуктан мындай маселелерди программалоодо C++ тилинде өзүнчө оператор колдонулат. Бул операторду тандоо оператору деп аташат.

Тандоо операторунун жалпы түрү. switch <туюнтма>
{case n1:m1; break; case n2:m2;
break;
. . .
case nk:mk; break;
default: m; break
}

Тандоо оператору эки бөлүктөн турат:-операторду*н башы* жана *тулкусу*.

башы switch <туюнтма> *тулкусу* {case n1:m1;
break; case n2:m2; break;
. . .
case nk:mk; break; default: m;
break
}

Тандоо операторунун толук жана толук эмес болуп эки түрү кезигет. Толук эмес түрүндө default: m; break сабы колдонулбайт. Бул жалпы берилиште <туюнтма>-бүтүн маанилүү туюнтма же символдук турактуулар болушу мүмкүн. n1, n2, …, nk-белгилери б.а. бүтүн туруктуулар же бүтүн маанилүү туюнтма же символдук турактуулар. m1, m2, …, mk-операторлор. Тандоо операторундагы кызматчы сөздөрдүн мааниси switch- бөлүштүрүү, case-тандоо, default-дал келбөө, break-токтоо.

Операторду*н* аткарылышы адегенде <туюнтманын> мааниси эсептелинет да анын мааниси n1, n2, …, nk турактуулары менен салыштырылат. Салыштыруу учурунда <туюнтманын> мааниси турактуулардын кайсы бирине дал келсе анда ошол турактуу көрсөткөн оператор аткарылат. *Мисалы:* n2 дал келсе m2 оператору аткарылат ж.б.у.с. Эгерде <туюнтманын> мааниси эч бир n1, n2, …, nk турактууларына дал келбесе, анда default кызматчы сөзүнөн кийин m деген оператор аткарылат.

Тандоо операторун колдонууда белгилерди (n1, n2, …, nk) ирээтеп жазуу талап кылынбайт, ошондой эле default кызматчы сөзү сөзсүз түрдө эле тандоочу операторду*н* аягында жазылбайт, тандоочу операторунун тулкусунун каалагандай жеринде жайгаша алат жана default сөзүн такыр эле колдонбой койсо деле болот. Бул учурда <туюнтманын> мааниси эч бир n1, n2, …, nk турактуулардын маанисине дал келбейт жана default сөзүнөн кийинки m деген оператор жок дегенди билдирет б.а. switch оператору эч кандай иш аракетти аткарбай калат. Бул бөлүштүрүү операторунун толук эмес түрүнө дал келет. Тандоо операторунда n1, n2, …, nk белгилери бирин-бири кайталабаган турактуулар болушу керек. Ал эми break (үзүлүү) оператору аткарылганда башкаруу switch-тандоо операторунун тулкусунан кийинки операторго берилет. Break оператору колдонулбай калган учуру да кезигет. Эгерде break оператору кандайдыр бир mi операторлордун тобунан кийин жазылбаса, анда mi, mi+1, mi+2,... ж.б.у.с. операторлору break кайрадан жолукканга чейин аткарылат. Жалпы жазылышы төмөндөгүдөй:

switch <туюнтма>
{case n1:m1; break; case n2:m2;
break;
. . .
case ni-1:mi-1; break; case ni:mi;

```
  case ni+1:mi+1;
     . . .
  case nk:mk; break; default: m;
  break;
}
```

Эгерде эки же андан көп иш аракеттер дал келишсе б.а. m1, m2, …, ml операторлору бирдей болсо алардын жалпы жазылышы төмөндөгүдөй болот.

```
switch <туюнтма>
{ case n1: case n2:
   . . .
   case nl:ml; break;
   . . .
   case nk:mk; break; default: m;
   break;
}
```

Жогорку тандоо операторуна мисал карайлы.

Мисал: Арифметикалык амалдарды аткаргыч программаны түзөлү.

```
# include <stdio.h> main ( )
 {float a,b; char z;
   printf ("\n Амал белгисинин оң жагындагы жана сол жагындагы сандарды жана белгинин
арасына ачык калтырбай киргиз:");
  scanf ("%f %c %f", &a,&z,&b); switch (z)
 { case '+':printf ("%f%c%f=%f\n",a,z,b,a+b); break; case '-':printf ("%f%c%f
    =%f\n", a,z,b,a-b); break; case '*':printf ("%f%c%f =%f\n", a,z,b,a*b); break;
    case '/':printf ("%f%c%f =%f\n", a,z,b,a/b); break; case '>':;
    case '<':;
    case '=':printf ("\n бул катыш амалы"); break; default:printf ("\n бул белгисиз
    амал");
  }
}
```

Жыйынтык: Амал белгисинин оң жагындагы жана сол жагындагы сандарды жана белгинин арасына ачык калтырбай киргиз:

3+5

3.000000+5.000000=8.000000

Амал белгисинин оң жагындагы жана сол жагындагы сандарды жана белгинин арасына ачык калтырбай киргиз:

9-7

9.000000-7.000000=2.000000

Амал белгисинин оң жагындагы жана сол жагындагы сандарды жана белгинин арасына ачык калтырбай киргиз:

5*7

5.000000*7.000000=35.000000

Амал белгисинин оң жагындагы жана сол жагындагы сандарды жана белгинин арасына ачык калтырбай киргиз:

9/5

9.000000/5.000000=1.800000

Амал белгисинин оң жагындагы жана сол жагындагы сандарды жана белгинин арасына ачык калтырбай киргиз:
6>2

бул катыш амалы

Амал белгисинин оң жагындагы жана сол жагындагы сандарды жана белгинин арасына ачык калтырбай киргиз:
5<7

бул катыш амалы

Амал белгисинин оң жагындагы жана сол жагындагы сандарды жана белгинин арасына ачык калтырбай киргиз:
8^9

бул белгисиз амал

Токтоочу оператор

Токтоочу операторду жалпы түрү: break;

Бул оператор кайталанууну же баяндоону токтотуу үчүн колдонулат. break оператору for, while, do, switch операторлорунун тулкусунда кезигип калса, анда кайталануу же тандоо иш аракеттери токтотулат. Токтоочу операторду кайталануучу жана тандоочу операторлордон башка жерлерде жазылышы кашаага алынып калат, ошондой эле башкы функциянын (main) тулкусунан чыгуу үчүн колдонууга болбойт.

1- Мисал. z -тамгасын киргизгенге чейинки, мурунку киргизилген тамгаларды чыгаруучу программа.

```
# include <stdio.h> main()
  {char ch;
   for (; ;) {ch=getchar(); if (ch= =’z’)
   break; printf (“%c”,ch);
         }
}
```

Жыйынтык:

a a b b z

Эгерде кирүү сабы abcdzefh түрүндө болсо z ке чейинки гана abcd маалыматы чыгарылат.

2- Мисал. 0 дөн 100 чейнки сандардын ичинен 100 дөн квадраттары кичине болгон сандарды чыгаруучу программа.

```
# include <stdio.h>
  main ()
   {int i ;
    printf (“ Сан          квадраты\n” for (i=0;
    i<100; i++)
    { if(i*i>=100)break;
    printf (“%d          %d\n”, i, i*i );


    }
  }
```

Жыйынтык :

Сан	квадраты
0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81

Улантуучу оператор.

Бул оператордун жалпы түрү:continue;

Улантуучу операторду кайталануу процессинин кандайдыр бир бөлүгүн калтырып, кийинки кайталануучу циклди аткарууну ишке ашырат. Эгерде кийинки кайталануу (цикл) жок болсо (түгөнсө) анда кайталануу иш аракеттери аяктайт. Демек continue оператору while жана do операторлорунда колдонулса өзүнөн кийин башкарууну while же do операторлорундагы шарттарды текшерүүгө берет, ал эми for операторунда болсо циклдин параметиринин өзгөрүү кадамына берилет. Мисалы: 3кө эселүү болгон 20 га чейинки сандарды чыгаруучу программа.

```
# include <stdio.h> main()
  {int i;
   for(i=1; i<20; i++)
   { if (i%3) continue ; printf ("%d\n",i);
   }
}
```

Жыйынтык:

3
6
9
12
15
18

Программага түшүндүрмө берели. Кайталануу 1 ден 20 чейин жүрөт. Эгерде санды 3 кө бөлүүнүн калдыгын табуучу (%) амал калдык 0 болсо шарттуу оператор if шартты же туюнтманы жалган деп эсептеп printf("%d\n",i); оператору аткарылат. Эгерде санды 3 кө бөлгөндө калдык 0 дөн айрмалуу болсо анда шарт же туюнтма чын деп эсептеп continue; оператору аткарылып, for операторунун циклинин кийнки кадамы аткарылат, мындай процесс i=20 болгонго чейин жүрөт.