«Қазіргі программалау технологиялары» курсы
(C# тілінде программалау)

4 лекция. Енгізу-шығару мысалдары. Басқару операторлары элементтеріне мысалдар

Консольдік енгізу-шығару (Console класы) мүмкіндіктері және басқару операторлары элементтері (тармақ, цикл, басқаруды беру).

Консольге мәлімет шығару

```
using System;
namespace A
{    class Class1
   {    static void Main()
      {
         int       i = 3;
         double  y = 4.12;
         decimal d = 600m;
         string    s = "Берік";
         Console.Write( i );
         Console.Write( " y = {0:F2} \nd = {1:D3}", y, d );
         Console.WriteLine( " s = " + s );
      }
   }
}
```

Программа жұмысы нәтижесі :

3 y = 4,12

d = 600 s = Берік

Консольден мәлімет енгізу

```
using System;
namespace A
{    class Class1
   {    static void Main()
      {
         string s = Console.ReadLine();          // ввод строки

         char c = (char)Console.Read();          // ввод символа
         Console.ReadLine();

         string buf;                             // буфер для ввода чисел
         buf = Console.ReadLine();
         int i = Convert.ToInt32( buf );         // преобразование в целое

         buf = Console.ReadLine();
         double x = Convert.ToDouble( buf ); // преобразование в вещ.

         buf = Console.ReadLine();
         double y = double.Parse( buf );         // преобразование в вещ.
      }
   }
}
```

Математикалық функциялар: Math класы

Аты	Сипаты	Нәтижесі	Түсініктеме
`Abs`	Модуль	асыра жүктелген	|x| → `Abs(x)`
`Acos`	Арккосинус	`double`	`Acos(double x)`
`Asin`	Арксинус	`double`	`Asin(double x)`
`Atan`	Арктангенс	`double`	`Atan(double x)`
`Atan2`	Арктангенс	`double`	`Atan2(double x, double y)`—y/x мәніне тең бұрыш тангенсі
`BigMul`	Көбейту	`long`	`BigMul(int x, int y)`
`Ceiling`	Үлкен бүтінге дейін дөңгелектеу	`double`	`Ceiling(double x)`
`Cos`	Косинус	`double`	`Cos(double x)`
`Cosh`	Гиперболалық косинус	`double`	`Cosh(double x)`
`DivRem`	Бөлу және қалдық	асыра жүктелген	`DivRem(x, y, rem)`
`E`	Натурал логарифм негізі (*e* саны)	`double`	2,71828182845905
`Exp`	Экспонента	`double`	e^x → `Exp(x)`

Floor	Кіші бүтінге дейін дөңгелектеу	double	Floor(double x)
IEEERemainder	Бөлгендегі қалдық	double	IEEERemainder(double x, double y)
Log	Натурал логарифм	double	$\log_e x$ → Log(x)
Log10	Ондық логарифм	double	$\log_{10} x$ → Log10(x)
Max	Екі сан максимумы	асыра жүктелу	Max(x, y)
Min	Екі сан минимумы	перегружен	Min(x, y)
PI	π саны мәні	double	3,14159265358979
Pow	Дәрежелеу	double	x^y → Pow(x, y)
Round	Дөңгелектеу	асыра жүктелу	Round(3.1) нәтижесі 3 Round (3.8) нәтижесі 4
Sign	Сан таңбасын табу	int	Аргументтері асыра жүктелген
Sin	Синус	double	Sin(double x)
Sinh	Гиперболалық синус	double	Sinh(double x)
Sqrt	Квадрат түбір	double	$\sqrt{x}$ → Sqrt(x)
Tan	Тангенс	double	Tan(double x)
Tanh	Гиперболалық тангенс	double	Tanh(double x)

Мысал: температураны Фаренгейттен (F) Цельсияға (C) көшіру программасы

$$C = \frac{5}{9}(F - 32)$$

```
using System;
namespace CA1
{    class Class1
  {    static void Main()
    {
      Console.WriteLine( "Введите температуру по Фаренгейту" );
      double fahr = Convert.ToDouble(Console.ReadLine() );

      double cels =5.0 / 9 * (fahr - 32);

      Console.WriteLine( "По Фаренгейту: {0} в градусах Цельсия: {1}",
                         fahr, cels );
    }
  }
}
```

Басқару операторларының элементтері

Блок (құрама оператор)

Блок — жүйелі жақшаларға алынған операторлар тізбегі:

```
begin   end
{       }
```

Блок компилятор үшін бір оператор болып саналады, **синтаксис бойынша бір оператор керек болғанмен, алгоритм бойынша — бірнеше оператор орындалады**.

Блокта бір-ақ оператор болуы мүмкін, кейде ол тіпті бос болады.

«Өрнек» операторы

- Нүктелі үтірмен аяқталған кез келген өрнек оператор болып есептеледі, ол белгілі бір амалдар орындауды керек етеді.

```
i++;            // выполняется операция инкремента
a *= b + c;     // выполняется умножение с присваиванием
fun( i, k );    // выполняется вызов функции
```

Бос оператор

- *Бос оператор* **;** синтаксис бойынша оператор керек болғанмен, мағынасы бойынша — ол қажет етілмегенде қолданылады:
- while (true);

Бос операторлардан тұратын бұл цикл шексіз орындалуды көрсетеді

- ;;;

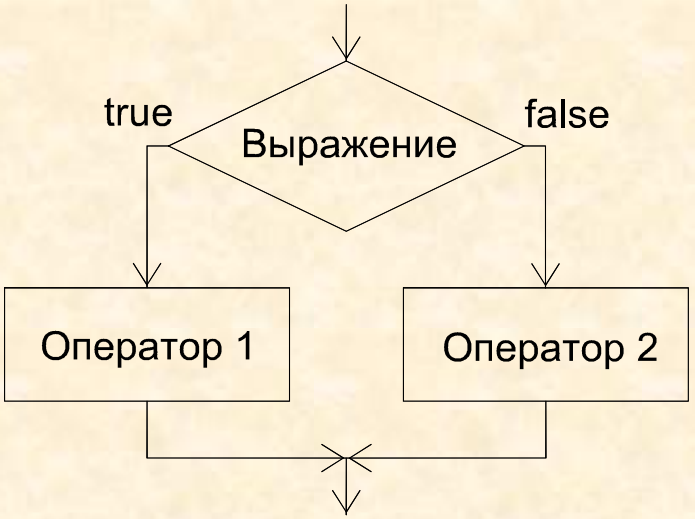
Үш бос оператор

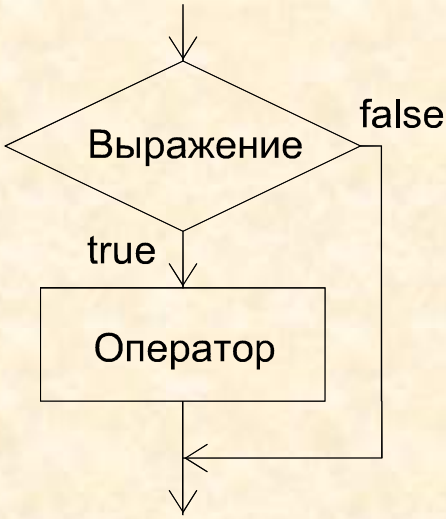
Тармақталу операторы:

- тармақталу (if)
- ауыстыру (switch)

if шартты операторы

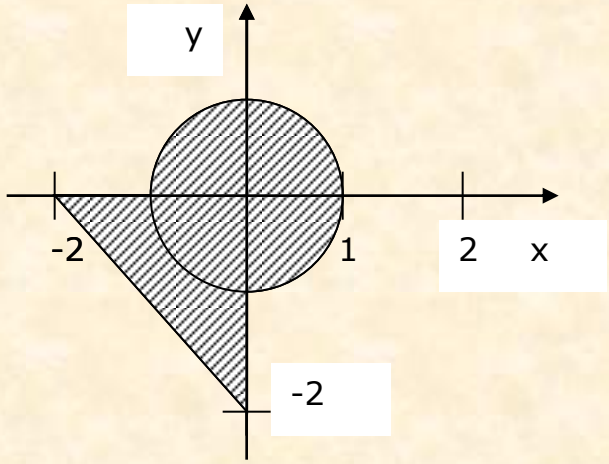
if (өрнек) оператор_1;
[else оператор_2;]






```
if ( a < 0 ) b = 1;
if ( a < b && (a > d || a == 0))   ++b;
else {  b *= a; a = 0; }
if ( a < b ) if ( a < c ) m = a;
                  else          m = c;
else              if ( b < c ) m = b;
                  else          m = c;
```

Мысал

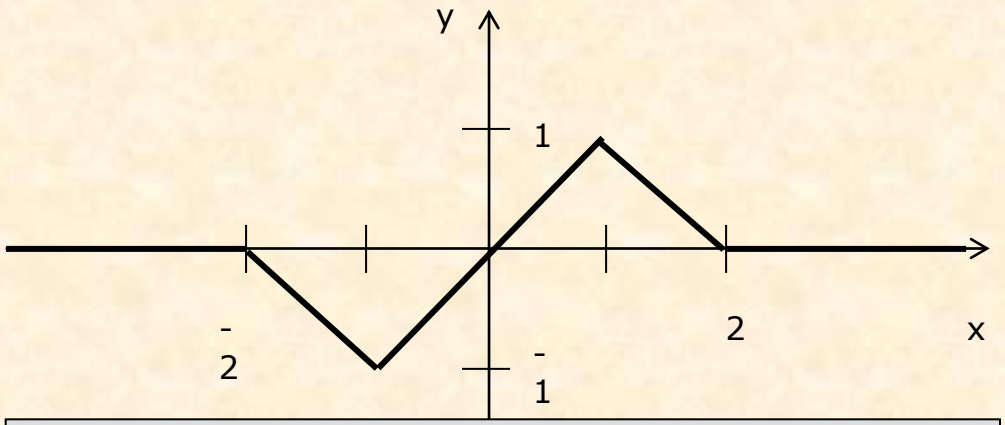



```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {
            Console.WriteLine( "Введите координату x" );
            double x = Convert.ToDouble(Console.ReadLine() );

            Console.WriteLine( "Введите координату y" );
            double y = double.Parse(Console.ReadLine() );

            if  ( x * x + y * y <= 1 ||
                  x <= 0 && y <= 0 && y >= - x – 2 )
                Console.WriteLine( " Точка попадает в область " );
            else
                Console.WriteLine( " Точка не попадает в область " );
}}}
```

2 мысал



$$y=\begin{cases}0, & x<-2\\ -x-2, & -2\le x<-1\\ x, & -1\le x<1\\ -x+2, & 1\le x<2\\ 0, & x\ge 2\end{cases}$$

```
if ( x < -2 )                  y = 0;
if ( x >= -2 && x < -1 ) y = -x - 2;
if ( x >= -1 && x <  1 ) y = x;
if ( x >=  1 && x <  2 ) y = -x + 2;
if ( x >=  2 )                 y = 0;
```

```
if      ( x <= -2 )  y = 0;
else if ( x < -1 )     y = -x - 2;
  else if ( x <  1 )     y = x;
    else if ( x <  2 )     y = -x + 2;
      else                   y = 0;
```

```
y = 0;
if ( x > -2 ) y = -x - 2;
if ( x > -1 ) y = x;
if ( x >  1 ) y = -x + 2;
if ( x >  2 ) y = 0;
```

Нақты сандарды теңдік арқылы тексеру

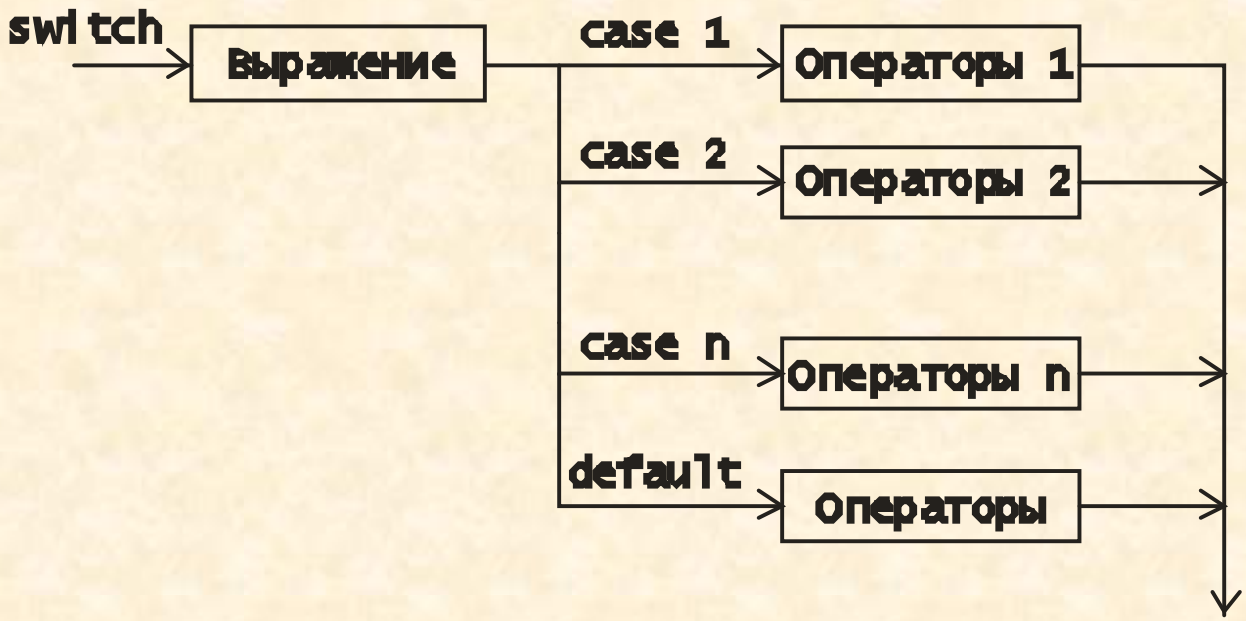
- Компьютер жадында нақты мәндерді бейнелеу дәлдіктеріне байланысты оларды теңдікте тексеру қажет емес, оның орнына айырма модулін бір кішкене санмен салыстыру керек.
- float a, b; …
- if (a == b) … // ұсынылмайды
- if (Math.Abs(a - b) < 1e-6) … // осылай сенімді!
- Айырма модулін салыстыру шамасының мәні есепке байланысты оның айнымалылары дәлдігіне қарай таңдалып алынады.
- Бұл шама төменнен Single және Double кластарында анықталған Epsilon константасымен шектеледі. Бұл айнымалының мүмкін ең кіші мәні мынадай шартқа сәйкес болуы тиіс:

  1.0 + Epsilon != 1.0

switch таңдау операторы

```
switch ( выражение ){
    case константное_выражение_1: [ список_операторов_1 ]
    case константное_выражение_2: [ список_операторов_2 ]

    case константное_выражение_n: [ список_операторов_n ]
    [ default: операторы ]
}
```

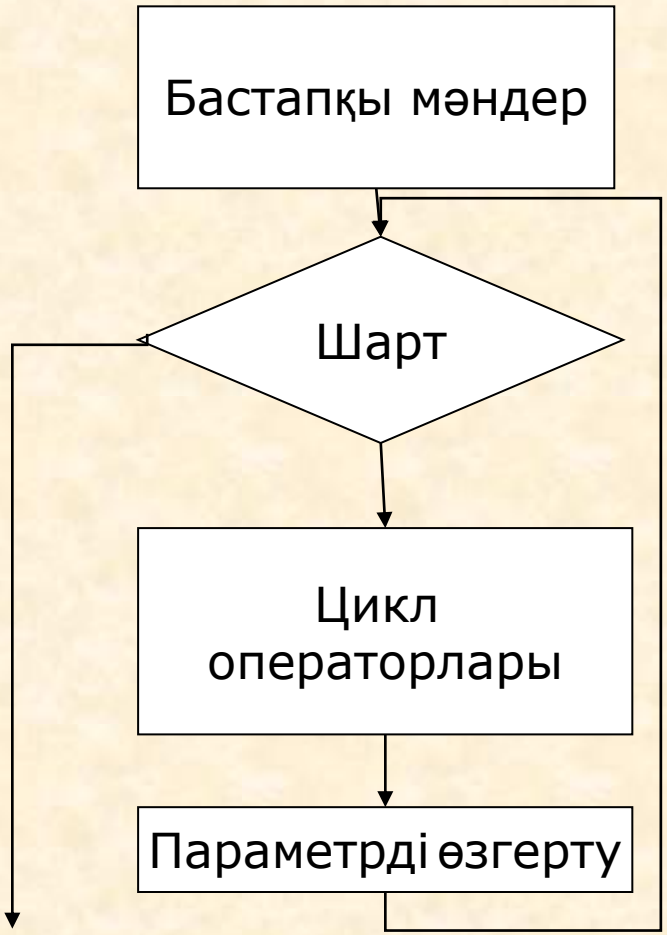


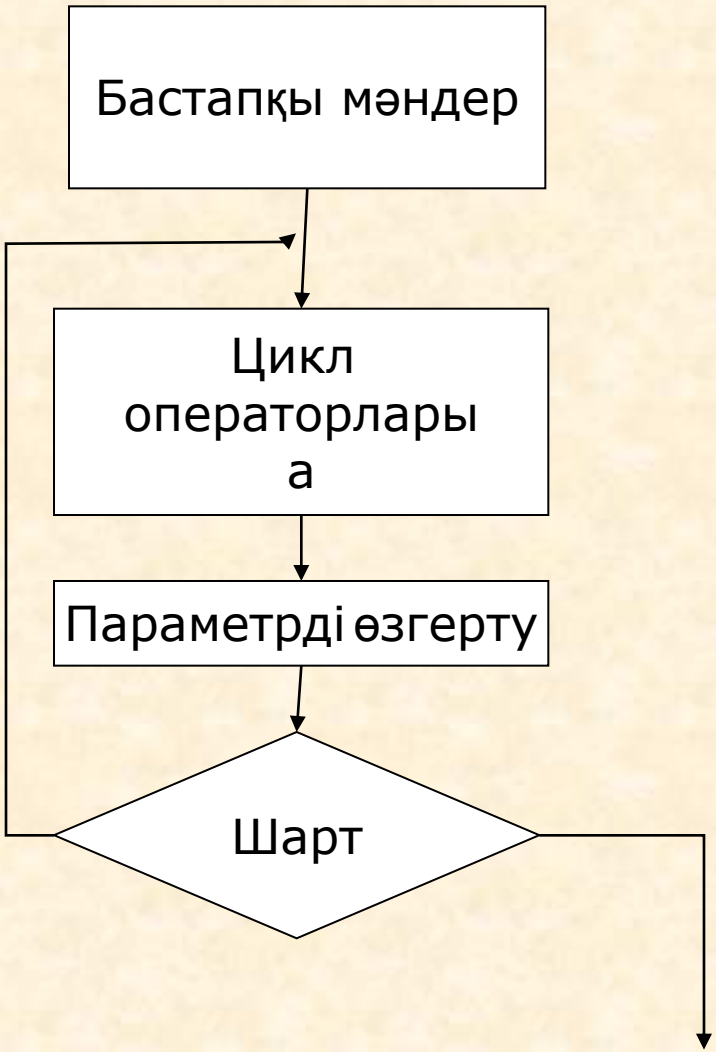
Мысал: 4 амал орындайтын калькулятор

```
using System; namespace ConsoleApplication1
{   class Class1    {   static void Main()       {
       string buf;  double a, b, res;
       Console.WriteLine( "Введите 1й операнд:" );
       buf = Console.ReadLine(); a = double.Parse( buf);
       Console.WriteLine( "Введите знак" );
       char op = (char)Console.Read(); Console.ReadLine();
       Console.WriteLine( "Введите 2й операнд:" );
       buf = Console.ReadLine(); b = double.Parse( buf);
        bool ok = true;
        switch (op)
           {
              case '+' : res = a + b; break;
              case '-'  : res = a - b;  break;
              case '*' : res = a * b; break;
              case '/' : res = a / b; break;
              default  : res = double.NaN; ok = false; break;
            }
       if (ok) Console.WriteLine( "Результат: " + res );
        else    Console.WriteLine( "Недопустимая операция" );
}}}
```

Цикл операторлары

Цикл операторлары құрылымы

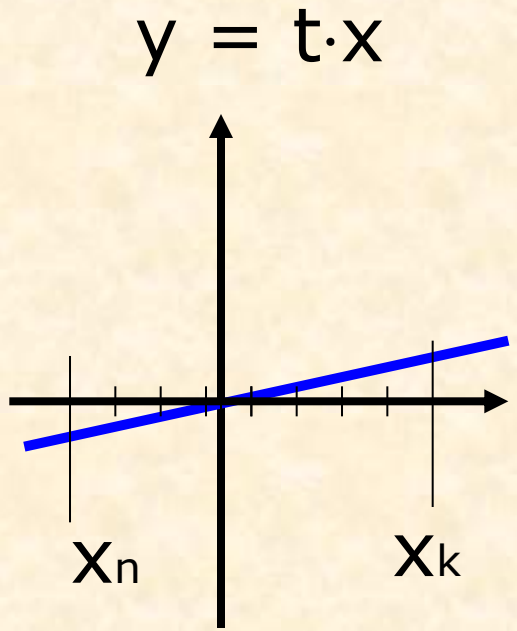




Алғы шартты цикл

while (өрнек) оператор

```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {
            double Xn = -2, Xk = 12, dX = 2, t = 2, y;
            Console.WriteLine( "|     x     |     y     |" );
            double x = Xn;
            while ( x <= Xk )
            {
                y = t * x;
                Console.WriteLine( "| {0,9} | {1,9} |", x, y );
                x += dX;
            }
        }
    }
}
```



Соңғы шартты цикл

do оператор while өрнек;

```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {
            char answer;
            do
            {
                Console.WriteLine( "Купи слоника, а?" );
                answer = (char) Console.Read();
                Console.ReadLine();
            } while ( answer != 'y' );
        }
    }
}
```

Параметрлі цикл

for (инициалдау; өрнек; модификациялау) оператор;

```
int s = 0;
for ( int i = 1; i <= 100; i++ ) s += i;
```

Параметрлі циклге мысал

```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {
            double Xn = -2, Xk = 12, dX = 2, t = 2, y;
            Console.WriteLine( "|     x     |     y     |";
            for ( double x = Xn; x <= Xk; x += dX )
            {
                y = t * x;
                Console.WriteLine( "| {0,9} | {1,9} |", x, y );
            }
        }
    }
}
```

Циклдерді құру ережелері

- while және for циклдерінің ішкі операторлары бірден артық болса, оларды блок түрінде жазу керек;
- цикл ішіндегі меншіктеу операторларының оң жағында көрсетілген айнымалылардың бәріне мән берілуі тиіс;
- цикл ішінде цикл шартына кіретін айнымалылардың (болмағанда біреуінің) мәні өзгертілуін қадағалау қажет;
- итерациялық циклдерден оның қадамдарының мүмкін болатын белгілі бір үлкен саны орындалған соң, одан авариялық түрде шығу қарастырылуы тиіс.

Басқаруды беру операторлары

Басқаруды беру (передача управления)

- break операторы — өзі ішінде орналасқан циклді аяқтайды;
- continue операторы — циклдің келесі қадамына көшуді орындайды;
- return операторы — өзі ішінде тұрған функциядан шығуды орындайды;
- throw операторы — ерекше жағдайды туындатады (генерирует исключительную ситуацию);
- goto операторы — шартсыз көшу ісін атқарады.

Мысал: қатар қосындысын есептеу

sin*x* функциясының мәнін берілген дәлдікпен (ε) дәрежелі көпмүшелік қатар арқылы төмендегідей формуламен есептеу программасын жазу керек:

$$y = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

$$|C_n| \le \varepsilon$$

$$C_n = (-1)^n \frac{x^{2n-1}}{(2n-1)!}$$

$$C_{n+1} = -C_n \frac{x^2}{2n\ (2n+1)}$$

Мысал: қатар қосындысын есептеу

```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {  double e = 1e-6;              const int iterLimit = 500;
           Console.WriteLine( "Введите аргумент:" );
           double x = Convert.ToDouble(Console.ReadLine());
           bool error = false;          // признак ошибки
           double c = x, y = c;         // член ряда и сумма ряда
           for ( int n = 1; Math.Abs(c) > e; n++ )
           {    c *= - x * x / ((2 * n ) * ( 2 * n + 1 ));
                y += c;
                if ( n > iterLimit ) { error = true; break; }
           }
           if ( error ) Console.WriteLine( "Ряд расходится" );
           else         Console.WriteLine( "Сумма ряда - " + y );
}}} end.
```

return операторы

завершает выполнение функции и передает управление в точку ее вызова:

return [выражение];

Оператор goto

goto метка;

В теле той же функции должна присутствовать ровно одна конструкция вида:

метка: оператор;

goto case константное_выражение;

goto default;

Ерекшеліктерді өңдеу (обработка исключений)

Исключительная ситуация, или исключение — это возникновение непредвиденного или аварийного события, которое может порождаться некорректным использованием аппаратуры.

НаМысал, это деление на ноль или обращение по несуществующему адресу памяти.

Исключения позволяют логически разделить вычислительный процесс на две части — обнаружение аварийной ситуации и ее обработка.

Возможные действия при ошибке

- прервать выполнение программы;
- возвратить значение, означающее «ошибка»;
- вывести сообщение об ошибке и вернуть вызывающей программе некоторое приемлемое значение, которое позволит ей продолжать работу;
- выбросить исключение

Исключения генерирует либо система выполнения, либо программист с помощью оператора throw.

Кейбір стандартты ерекше жағдайлар

Аты	Түсініктемесі
`ArithmeticException`	Ошибка в арифметических операциях или преобразованиях (является предком `DivideBeZeroException` и `OverFlowException`)
`DivideByZeroException`	Попытка деления на ноль
`FormatException`	Попытка передать в метод аргумент неверного формата
`IndexOutOfRangeException`	Индекс массива выходит за границы диапазона
`InvalidCastException`	Ошибка преобразования типа
`OutOfMemoryException`	Недостаточно памяти для создания нового объекта
`OverFlowException`	Переполнение при выполнении арифметических операций
`StackOverFlowException`	Переполнение стека

try операторы

Служит для обнаружения и обработки исключений.

Оператор содержит три части:

- *контролируемый блок* — составной оператор, предваряемый ключевым словом try. В контролируемый блок включаются потенциально опасные операторы программы. Все функции, прямо или косвенно вызываемые из блока, также считаются ему принадлежащими;
- один или несколько *обработчиков исключений* — блоков catch, в которых описывается, как обрабатываются ошибки различных типов;
- *блок завершения* finally, выполняемый независимо от того, возникла ли ошибка в контролируемом блоке.

Синтаксис оператора try:

try блок [catch-блоки] [finally-блок]

Ерекше жағдайларды өңдеу механизмдері (обработка исключений)

- Обработка исключения начинается с появления ошибки. Функция или операция, в которой возникла ошибка, генерируют исключение;
- Выполнение текущего блока прекращается, отыскивается соответствующий обработчик исключения, и ему передается управление.
- В любом случае (была ошибка или нет) выполняется блок finally, если он присутствует.
- Если обработчик не найден, вызывается стандартный обработчик исключения.

1 мысал:

```
try {

                        // Контролируемый блок
}
catch ( OverflowException e ) {

                        // Обработка переполнения
}
catch ( DivideByZeroException ) {

                        // Обработка деления на 0
}
catch {
        // Обработка всех остальных исключений
}
```

2 мысал: проверка ввода

```
static void Main()
        {
            try
            {
                Console.WriteLine( "Введите напряжение:" );
                double u = double.Parse( Console.ReadLine() );
                Console.WriteLine( "Введите сопротивление:" );
                double r = double.Parse(Console.ReadLine() );
                double i = u / r;
                Console.WriteLine( "Сила тока - " + i );
            }
            catch ( FormatException )
            {
                Console.WriteLine( "Неверный формат ввода!" );
            }
            catch                                   // общий случай
            {
                Console.WriteLine( "Неопознанное исключение" );
            }
        }
```

throw операторы

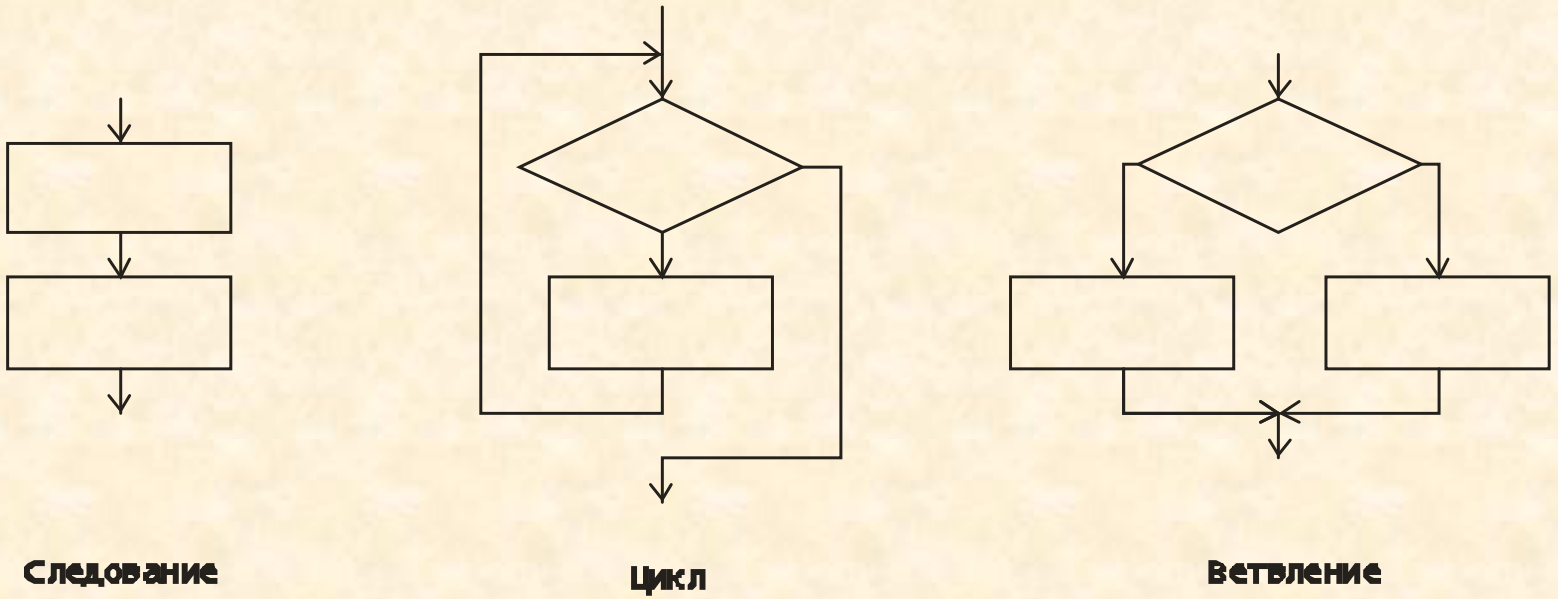
- **throw [выражение];**

Параметр должен быть объектом, порожденным от стандартного класса System.Exception. Этот объект используется для передачи информации об исключении его обработчику.

Мысал:

```
throw new DivideByZeroException();
```

Құрылымдық программалаудың негізгі конструкциялары



- Целью использования базовых конструкций является получение программы простой структуры. Такую программу легко читать, отлаживать и при необходимости вносить в нее изменения.
- Особенностью базовых конструкций является то, что любая из них имеет только один вход и один выход, поэтому конструкции могут вкладываться друг в друга