# **Каталар менен иштөө**

## **Өзгөчө абалдарды иштетүү**

Си++ тилинде каталар менен иштөөгө багытталган атайын механизми бар. Аны ***өзгөчө абалдарды иштетүү*** деп аталат. Бул механизмдин жардамы менен програм аткарылып жаткан учурда башкарууну жөнөкөйлөтүп тиешелүү таасир берсе болот. Өзгөчө абалдарды иштетүүнү ***try, catch***жана***throw*** кызматчы сөздөр аркылуу жүргүзүлөт. Жалпы сөз менен айтканда өзгөчө абалдарды иштетүү бойунча көрсөтмөлөр ***try*** блогунда жайгашат. Эгерде өзгөчө абал (ката) ***try*** блок ичинде аныкталса, ал козголот (***throw*** кызматчы сөзү)***, catch***блогунда кармалат жана иштетилет.

Өзгөчө абалды козгогон каалаган көрсөтмө ***try*** блогунун ичинде аткарылыш керек. Каалагандай өзгөчө абал ***try*** блогунан кийин жайгашкан ***catch***көрсөтмө аркылуу кармалыш керек. Төмөндө ***try***жана ***catch***көрсөтмөлөрдүн негизги формалары келтирилген:

try  
{  
// Өзгөчө абалды козгогон блок  
}  
  
catch(тип\_1 арг)  
{// Өзгөчө абалды кармаган блок  
}  
  
catch(тип\_2 арг)  
{// Өзгөчө абалды кармаган блок  
}  
  
catch(тип\_N арг)  
{// Өзгөчө абалды кармаган блок  
}

Өзгөчө абал козголгондон кийин ал конкреттүү катага жараша тиешелүү ***catch***көрсөтмө аркылуу кармалат. ***try*** блогу менен бир нече ***catch***көрсөтмө байланышта турушу мүмкүн. ***catch***көрсөтмөнүн кайсынысы колдоноору өзгөчө абалдан көз каранды болот, б.а. ***catch***көрсөтмөдө көрсөтүлгөн аргументтин тиби өзгөчө абалдын тибине дал келгени иштейт. Өзгөчө абал кармалган учурда ***арг*** аргументи ошол өзгөчө абалдын маанисин алат. Эгерде өзгөчө абалдын өзү кереги жок болсо, анда ***catch***көрсөтмөдө анын тибин гана көрсөтүп ***арг*** аргументин көрсөтпөй койсо болот. Негизинен каалаган типтеги берилиштерди жана түзүлгөн класс тибиндеги берилиштерди кармап алса болот.

Төмөндө ***throw*** көрсөтмөсүнүн негизги калыбы берилген:

throw өзгөчө\_абал;

***throw*** көрсөтмөсү ***try*** блогунун ичинде же бул блок чакырган каалаган функциянын ичинде аткарылышы мүмкүн. Мында ***өзгөчө\_абал****throw* көрсөтмөсү аркылуу козголуп жаткан өзгөчө абал.

Эгерде биз козгогон өзгөчө абал үчүн тиешелүү ***catch***көрсөтмөсү болбосо, анда програм ката менен иштөөсүн айактап калышы мүмкүн. Иштетилбеген өзгөчө абалды козгоо китепканалык ***terminate()*** стандарттуу функциянын чакырылышына алып келет. Унчукпаганда програмды айактоо үчүн ***terminate()*** функциясы ***abort()*** функциясын чакырат. Бирок анын ордуна колдонуучу өзүнүн функциясын түзүп койсо деле болот.

Төмөнкү програмда Си++ тилинде өзгөчө абалдарды иштетүү кандайча ишке ашырылаары көрсөтүлгөн.

#include <iostream>  
using namespace std;  
  
int main()  
{  
    cout<<"Програм башы\n";  
    try  
    { cout<<"try блогунун ичинде\n";  
        throw 10;// өзгөчө абалды козгоо  
        cout<<"Бул сап аткарылбайт\n";  
    }  
    catch (int i)  
    {//өзгөчө абалды кармоо  
        cout<<i<<" номерлүү ката кармалды\n";  
    }  
    cout<<"Програм айагы\n";  
        return 0;  
}

Програм төмөнкү жыйынтыкты берет:

Програм башы  
try блогунун ичинде  
10 номерлүү ката кармалды  
Програм айагы

Програмда үч көрсөтмөнү камтыган *try* блогу жана бүтүн типтеги өзгөчө абалды иштеткен *catch(int i)* көрсөтмөсү бар. *try* блогунун ичинде *cout* жана *throw* көрсөтмөлөрү гана аткарылат. Өзгөчө абал козголгондон кийин башкаруу *catch* блогуна берилип, *try* блогу өз ишин айактайт. Бул жерде *catch* көрсөтмөсү айкын эмес чакырылат дагы, башкаруу бул түзүмгө берилет. Демек *throw* көрсөтмөдөн кийин жайгашкан *cout*көрсөтмөсү аткарылбайт.

*catch* блогу аткарылгандан кийин башкаруу бул блоктон кийин турган көрсөтмөгө берилет. Бирок кээ бир учурларда *catch* блогу програмдын иштөөсүн эрксиз бүтүргөн *exit(), abort()* же кандайдыр бир башка функцияны чакыруу менен айактайт.

Мурда айтылгандай, өзгөчө абалдын тиби *catch* көрсөтмөсүнүн тибине дал келиш керек. Эгерде жогорку програмда *catch* көрсөтмөсүндө типти *double* тибине алмаштырсак, анда өзгөчө абал кармалбайт. Бул жобо төмөнкү програмда көрсөтүлгөн.

#include <iostream>  
using namespace std;  
  
int main()  
{  
    cout<<"Програм башы\n";  
    try  
    { cout<<"try блогунун ичинде\n";  
        throw 10;// өзгөчө абалды козгоо  
        cout<<"Бул сап аткарылбайт\n";  
    }  
    catch (double i)  
    {//Бул көрсөтмө аткарылбайт  
        cout<<i<<" номерлүү ката кармалды\n";  
    }  
    cout<<"Програм айагы\n";  
        return 0;  
}

Бүтүн типтеги өзгөчө абал *double* тибиндеги *catch* көрсөтмөсү аркылуу кармалбагандыктан, экранга төмөнкү саптар чыгат:

Програм башы  
try блогунун ичинде  
Abnormal program termination

Өзгөчө абал *try* блогуна кирбеген көрсөтмө аркылуу козголушу мүмкүн. Бирок бул учурда өзгөчө абалды козгогон көрсөтмө *try* блогу тарабынан чакырылган функциянын ичинде жайгашыш керек. Төмөнкү мисал мындай учурду сүрөттөйт.

#include <iostream>  
using namespace std;  
  
void func(int misal)  
{cout<<"Функциянын ичинде misal="<<misal<<'\n';  
    if(misal) throw misal;  
    }  
     
int main()  
{  
    cout<<"Програм башы\n";  
    try  
    { cout<<"try блогунун ичинде\n";  
        func(0);  
        func(1);  
        func(2);  
    }  
    catch (int i)  
    {  
        cout<<i<<" номерлүү ката кармалды\n";  
    }  
    cout<<"Програм айагы\n";  
        return 0;  
}

Програм төмөнкү жыйынтыкты берет:

Програм башы  
try блогунун ичинде  
Функциянын ичинде misal=0  
Функциянын ичинде misal=1  
1 номерлүү ката кармалды  
Програм айагы

*try* блогун функциянын ичинде жайгаштырса болот. Мындай учурларда функцияга ар бир кайрылууда өзгөчө абалды иштеткен көрсөтмө кайрадан калыбына келтирилет. Мисалы,

#include <iostream>  
using namespace std;  
  
void func(int misal)  
{  
    try  
    {  
        if(misal) throw misal;  
    }  
    catch (int i)  
    {  
        cout<<i<<" номерлүү ката кармалды\n";  
    }  
 }  
  
 int main()  
{  
    cout<<"Програм башы\n";  
        func(1);  
        func(2);  
        func(0);  
        func(3);  
    cout<<"Програм айагы\n";  
        return 0;  
}

Жыйынтык төмөнкү болот:

Програм башы  
1 номерлүү ката кармалды  
2 номерлүү ката кармалды  
3 номерлүү ката кармалды  
Програм айагы

Белгилеп кеткендей *try* блогу менен бир нече *catch* көрсөтмөсүн байланыштырса болот. Мындай учурларда ар бир *catch* көрсөтмөсү өзүнүн тибиндеги өзгөчө абалдарды кармоого багытталат. Төмөнкү мисалда бүтүн жана сап тибиндеги өзгөчө абалдар кармалат.

#include <iostream>  
using namespace std;  
  
void func(int misal)  
{  
    try  
    {  
        if(misal) throw misal;  
            else throw "Мааниси нөлгө барабар";  
    }  
    catch (int i)  
    {  
        cout<<i<<" номерлүү ката кармалды\n";  
    }  
    catch (char \*sap)  
    {  
        cout<<sap<<" сабы кармалды\n";  
    }  
     
 }  
  
 int main()  
{  
    cout<<"Програм башы\n";  
        func(1);  
        func(2);  
        func(0);  
        func(3);  
    cout<<"Програм айагы\n";  
        return 0;  
}

Экранда төмөнкү жыйынтык алабыз:

Програм башы  
1 номерлүү ката кармалды  
2 номерлүү ката кармалды  
Мааниси нөлгө барабар сабы кармалды  
3 номерлүү ката кармалды  
Програм айагы

Мында ар бир *catch* көрсөтмөсү тиешелүү типтеги өзгөчө абалдарды кармады.

Негизинен *catch* көрсөтмөлөрү програмда жайгашуу иретинде текшерилет. Тиби бойунча биринчи дал келген көрсөтмө аткарылып, калгандары көңүлгө алынбайт.

## **Өзгөчө абалдарды иштетүү бойунча кошумча маалыматтар**

Өзгөчү абалдарды иштетүү тутумунда бир катар аспектер бар. Кээ бир учурларда тибине көз карандысыз бардык өзгөчө абалдарды кармоого туура келет. Бул маселени чечүү үчүн *catch* көрсөтмөсүнүн төмөнкү калыбын колдонсо болот:

catch (…)  
{//бардык өзгөчө абалдарды иштетүү  
}

Мында көп чекит берилиштердин каалаган тибине туура келет. Төмөнкү програмда *catch (…)* көрсөтмөсүнүн иштөөсү сүрөттөлгөн.

#include <iostream>  
using namespace std;  
  
void func(int misal)  
{  
    try  
    {//int тибиндеги өзгөчө абалды козгоо  
        if(misal==0) throw misal;  
    //char тибиндеги өзгөчө абалды козгоо  
        if(misal==1) throw 'a';  
    //double тибиндеги өзгөчө абалды козгоо  
        if(misal==2) throw 45.77;  
    }  
    catch (...)  
    {  
        cout<<"Ката кармалды\n";  
    }  
  }  
  
 int main()  
{  
    cout<<"Програм башы\n";  
        func(1);  
        func(2);  
        func(0);  
        func(3);  
    cout<<"Програм айагы\n";  
        return 0;  
}

Экранга төмөнкү саптар чыгарылат:

Програм башы  
Ката кармалды  
Ката кармалды  
Ката кармалды  
Програм айагы

Үч типтеги өзгөчө абалдар бир *catch (…)* көрсөтмөсү аркылуу кармалды.

*catch (…)* көрсөтмөсүн *catch*көрсөтмөлөр тобунун айагына жайгаштыруу ыңгайлуу болот. Бул учурда *catch (…)* көрсөтмөсү унчукпаганда «барын кармаган» көрсөтмө болуп калат. Төмөнкү програмда бүтүн типтеги өзгөчө абалдар айкын кармалып, калган типтеги өзгөчө абалдар *catch (…)* көрсөтмөсү аркылуу кармалат.

#include <iostream>  
using namespace std;  
  
void func(int misal)  
{  
    try  
    {//int тибиндеги өзгөчө абалды козгоо  
        if(misal==0) throw misal;  
    //char тибиндеги өзгөчө абалды козгоо     
        if(misal==1) throw 'a';  
    //double тибиндеги өзгөчө абалды козгоо     
        if(misal==2) throw 45.77;  
    }  
    catch (int i)//int тибиндеги өзгөчө абалды кармоо  
    {  
        cout<<i<<" катасы кармалды\n";  
    }  
    catch (...)//каалаган типтеги өзгөчө абалды кармоо  
    {  
        cout<<"Ката кармалды\n";  
    }  
  }  
  
 int main()  
{  
    cout<<"Програм башы\n";  
        func(1);  
        func(2);  
        func(0);  
        func(3);  
    cout<<"Програм айагы\n";  
        return 0;  
}

Жыйынтык төмөнкү болот:

Програм башы  
Ката кармалды  
Ката кармалды  
0 катасы кармалды  
Програм айагы

*try* блогунан чакырылган функциялар үчүн өзгөчө абалдардын типтерин чектесе болот. Ал үчүн *throw* көрсөтмөсүн төмөнкү калыпта колдонуш керек:

кайтаруучу\_тип функция\_ысымы(арг\_тизмеси) throw (тип\_тизмеси)  
{  
//…  
}

Мында *тип\_тизмеси* функция тарабынан козголуучу типтердин тизмеси ал үтүр менен ажыратылып берилет. Тизмеден башка типтеги өзгөчө абалдар козголсо, анда програм кыйроо менен өз ишин айактайт. Эгерде функция эч кандай өзгөчө абалдарды козгобосун десек, анда *тип\_тизмеси*талаасын бош калтыруу керек.

Төмөнкү програмда өзгөчө абалдардын типтерин чектөөгө мисал келтирилген.

#include <iostream>  
using namespace std;  
  
void func(int misal) throw (int, char, double)  
{  
   //int тибиндеги өзгөчө абалды козгоо  
        if(misal==0) throw misal;  
    //char тибиндеги өзгөчө абалды козгоо     
        if(misal==1) throw 'a';  
    //double тибиндеги өзгөчө абалды козгоо     
        if(misal==2) throw 45.77;  
     
}  
  
int main()  
{  
    cout<<"Програм башы\n";  
    try{  
        func(0);//1 жана 2 маанилери да берип көргүлө  
        }  
    catch(int i){cout<<"int тиби\n";}  
    catch(char c){cout<<"char тиби\n";}  
    catch(double d){cout<<"double тиби\n";}  
    cout<<"Програм айагы\n";  
        return 0;  
}

Бул програмда *func()* функциясы *int***,***char*жана *double* тибиндеги өзгөчө абалдарды козгой алат. Башка типтеги өзгөчө абалды козгоо аракети жазалса, анда програм апаат менен өз ишин айактайт. Тастыктоо үчүн тизмеден *int* тибин алып таштап, програмды кайрадан аткартып көрүү керек.

*func()* функциясына төмөнкүдөй өзгөртүү киргизсек, анда каалагандай өзгөчө абалды козгоого тыйуу салынат.

void func(int misal) throw ()  
{  
   //int тибиндеги өзгөчө абалды козгоо  
        if(misal==0) throw misal;  
    //char тибиндеги өзгөчө абалды козгоо     
        if(misal==1) throw 'a';  
    //double тибиндеги өзгөчө абалды козгоо     
        if(misal==2) throw 45.77;  
    }

Өзгөчө абалды кайрадан козгосо болот. Анын мааниси өзгөчө абалды иштетүүгө бир нече процедурга берүүдө турат. Мисалы, бир процедур өзгөчө абалдын бир аспектисин, экинчиси башка аспектисин иштетсин дейли. Кайрадан өзгөчө абал *catch* блогунун ичинде гана же ал блоктон чакырылган каалаган функцияда козголушу мүмкүн. Өзгөчө абал кайрадан козголгондо ал ошол *catch* көрсөтмөсү аркылуу кармалбай, бул көрсөтмөгө карата башка, сырткы көрсөтмөгө өтөт. Төмөнкү програмда өзгөчө абал *char* тибине карата кайрадан козголот.

#include <iostream>  
using namespace std;  
  
void func()  
{  
    try{  
    //char тибиндеги өзгөчө абалды козгоо  
        throw "Salam";  
    }  
    //char тибиндеги өзгөчө абалды кармоо  
    catch(char \*){  
        cout<<"char тибиндеги өзгөчө абалды func() ичинде кармоо\n";  
    //char тибиндеги өзгөчө абалды кайрадан козгоо  
        throw;  
    }  
}  
  
   int main()  
{  
    cout<<"Програм башы\n";  
    try{  
        func();  
        }  
    catch(char \*){  
        cout<<"char тибиндеги өзгөчө абалды main() ичинде кармоо\n";}  
     cout<<"Програм айагы\n";  
        return 0;  
}

Програм төмөнкү жыйынтыкты берет:

Програм башы  
char тибиндеги өзгөчө абалды func() ичинде кармоо  
char тибиндеги өзгөчө абалды main() ичинде кармоо  
Програм айагы

## **new оператору козгогон өзгөчө абалдарды иштетүү**

Эсти бөлүүдө ийгиликтүү натыйжа алынбаса *new* оператору өзгөчө абалды козгойт. Си++ тилинин стандартына ылайык мындай учурда *new* оператору *bad\_alloc* өзгөчө абалын козгойт. Мындай өзгөчө абалды кармоого мүмкүндүк жок болсо програм апаат менен токтойт. Аталган өзгөчө абалга кирүүгө мүмкүндүк алуу үчүн програмга **<***new***>** бөрк файлын кошуу зарыл.

Си++ тилинин стандартында эсти бөлүүдө ката пайда болсо *new* оператору нөл маанисин кайтарып, өзгөчө абалды козгобойт. Төмөндө *new* операторунун негизги калыбы келтирилген:

көрсөткүч=new(nothrow) тип;

Мында *көрсөткүч*көрсөткүчү *тип* тибиндеги өзгөрмөгө көрсөткүч болот. *nothrow* кызматчы сөзү өзгөчө абалдын козголушун четке кагат.

*new*операторун колдонгон төмөнкү програмда *try/catch* блогу эсти бөлүүдө жол болбогон учурду текшерүүгө мүмкүндүгүн берет.

#include <iostream>  
#include <new>  
using namespace std;  
  
   int main()  
{int \*p;  
    try{  
        p=new int;//бүтүнгө эс бөлүү  
    }  
    catch(bad\_alloc kata)  
        {  
        cout<<"Эс бөлүүдө ката бар\n";  
            return 1;}  
     for(\*p=0;\*p<10;(\*p)++)  
         cout<<\*p<<' ';  
     delete p;//Эсти бошотуу  
        return 0;  
}

Мында эсти бөлүүдө ката кетип калса, анда ал ката *catch*көрсөтмөсү аркылуу кармалат.

Жогорку програмда эсти бөлүүдө ката кетишинин ыктымалдыгы өтө аз болгон үчүн, төмөнкү програмда андай абалды күчкө салуу менен козгойбуз. Эсти бөлүү процесси ката пайда болгонго чейин улана берет.

#include <iostream>  
#include <new>  
using namespace std;  
  
   int main()  
{double \*p;  
    do{  
        try{  
        p=new double[10000];  
         }  
    catch(bad\_alloc kata)  
        {  
        cout<<"Эс бөлүүдө ката бар\n";  
            return 1;}  
        cout<<"Эс бөлүүдө ката жок\n";  
        }while(p);  
    return 0;  
}

Төмөнкү мисалда *new* операторунун алтернативдүү *new (nothrow)* операторунун колдонулушу каралган. Жогорку програмдын кайрадан иштетилген версиясы.

#include <iostream>  
#include <new>  
using namespace std;  
  
   int main()  
{double \*p;  
    do{//цикл эсте орун калбаганча иштейт  
        p=new double[10000];  
        if(p) cout<<"Эс бөлуүүдө ката жок\n";  
            else  
            cout<<"Эс бөлүүдө ката бар\n";  
                     
    } while(p);  
    return 0;  
}

Програмда *new*операторун *nothrow* кызматчы сөзү менен колдонгондо эс бөлүү учурунда ар бир суроо-талаптан кийин көрсөткүчтүн кайтарып берген мааниси текшерилет.

## **КӨНҮГҮҮЛӨР**

Абстрак классты камтыган програм түзүп анын ичинде төмөнкүлөрдү ишке ашыргыла:

·         конструктор,

·         деструктор,

·         объектинин учурдагы абалын көрсөткөн korsot() виртуал функциясы,

·         тиешелүү эсептөөлөрдү жүргүзгөн esepte() функциясы.

Стандарттүү файлдык агымдарды колдонуп объектер жөнүндөгү маалыматтарды файлга сактагыла. Берилиштерди киргизүү/чыгаруу учурунда пайда болгон каталарды (мисалы файлды ачууга мүмкүн болбосо же дискте орун жетишсиз болсо) өзгөчө абалдарды иштетүү механизмин колдонгула.

1.       Китептер(Аты, Автору, жанр, жыл, барак саны, тираж, басылган барактардын саны (esepte() функциясы).

2.       Унаа (Аты, тиби, чыккан жылы, макс. ылдамдыгы, мотор көлөмү, бензин расходу, бензобактын көлөмү, бир бак менен жүрүүчү аралык (esepte() функциясы).

3.       Тамак аш азыктары (Аты, дүкөндүн бөлүмү, чыгарылган күнү, сактоо мөөнөтү, сактоо мөөнүтүнүн акыркы күнү(esepte() функциясы), салмак).

4.       Студент (Аты-жөнү, өткөн жылы, курс, алган сабактары, баалары, орточо балл (esepte() функциясы).

5.       Кыймылсыз мүлктөр (Дареги, тиби, кабат саны, бир кабаттагы батирлердин саны, подъезд саны, батирдин саны (esepte() функциясы).

6.       Спортсмендер (Аты-жөнү, спорт түрү, разряд, туулган күнү, медал саны, жаш категориясы (esepte() функциясы).

7.       Мезгил-мезгили менен чыгуучу басма сөз (Аты, тиби, бет саны, чыгуу саны, тиражы, бир жылды чыгуу саны (esepte() функциясы).

8.       Кадрлар бөлүмү (Аты-жөнү, бөлүм, кызматы, ишке кабыл алуу күнү, ички стажы (esepte() функциясы).

9.       Илим-изилдөө иштери (Аты, баштоо күнү, айактоо күнү (esepte() функциясы), изилдөө аймагы, кызматкерлердин саны, кызматкерлердин аты-жөнү.

10.    Програм жабдуулары(Аты, тиби, диск саны, ортнотуудан кийинки (толук, минималдуу, стандартту) көлөмү, кысуу пайызы (esepte() функциясы).

11.    ЭЭмдин бөлүктөрү (Аты, тиби, модели, баасы, саны, толук баасы (esepte() функциясы).

12.    Жүк ташуу (Жеткирүү пункту, аралык, унаа саны, унаалардын мамлекеттин номерлери [], буйум аттары [], чыгуу убактысы, келүү убактысы, жолго кеткен убакыт, орточо ылдамдык(esepte() функциясы).

13.    Аудио-студиясы (Аткаруучу/топ, аткаруучулар саны, стили, альбом саны, дискти жазуу баасы [], дисктин баасы [], тиражы [], аткаруучу/топтун жалпы кирешеси (esepte() функциясы).

14.    Чөнтөк телефондор (Аты, фирма, байланыш стандарты, аккумулятордун заряды, күтүү абалда сарпталуу, сүйлөө абалда зарпталуу, күтүү абалдагы убакыт (esepte() функциясы), сүйлөө абалдагы убакыт(esepte\_1() функциясы).

15.    Тармак жабдуулары (Аты, берилиштерди берүү ылдамдыгы, тиби, баасы, саны, берүү ылдамдыгы (байт/сек), жалпы баасы (esepte() функциясы).