# Osztály

*Az objektum egy azonosítható entitás, amely bizonyos jellemzőkkel, és viselkedéssel rendelkezik. Amikor egy osztály definiálva van akkor nincs lefoglalva memória, de amikor létrehozunk egy objetumot, akkor a memória lefoglalódik neki.*

## Felépítése

Egy class felépítésekor, meg kell adni annak nevét, változóit, lekérdező- és beállítómetódusait és konstruktorait.

|  |  |
| --- | --- |
| Ahhoz, hogy elkészítsünk egy osztály, csinálnunk kell, egy class objektumot (itt kell megadni az objektum nevét). Ezen belül kell felépítenünk egy osztály.  3 különböző láthatósági szintet tudunk megkülönböztetni, ebből 2-őt kell tudni a ZH-ra. Ez a **public,** és a **private.** Mi is a különbség a kettő között? A **public**-ba tartoznak azok a változók, amiket az osztályon kívülről tudunk elérni, és módosítani. A **private-**ba írt változókat viszont, csak az osztályon belül érjük el.  Ha egy változó értékét szeretnénk lekérdezni, akkor a **GET-t**, tehát a lekérdezőmetódust kell meghívni. Ha viszont módosítani szeretnénk az adatot, akkor a **SET**-t, azaz a beállítómetódust kell használni.  Végül meg kell írnunk a **destruktort**, ami megsemmisíti a konstruktor által létrehozott osztályobjektumokat. A destruktor neve megegyezik az osztály nevével amit egy ~ jel előz meg. | **CLASS** osztaly\_neve  {  **PUBLIC**:  osztaly\_neve() {}  **PRIVATE**:  INT **GET**VALTOZO\_1**() CONST**  **{**  **}**  **VOID** **SET**VALTOZO\_1**()**  **{**  **}**  **~**OSZTALY\_NEVE**() =** DEFAULT**;**  **};** |

## Konstruktor létrehozása

*A konstruktorok speciális osztálytagok, amelyeket a fordító minden alkalommal meghív, amikor az adott osztályhoz tartozó program példányosodik. A konstruktort minden esetben az osztályon belül a* ***PUBLIC****: részbe kell beírni.*

### Alapértelmezett konstruktor

Ha csak simán meg szeretnénk adni, hogy alapértelmezett (nulla) értéket vegyenek fel az objektumok, akkor ezt kétféleképpen tudjuk megtenni.

|  |  |
| --- | --- |
| Abban az esetben használjuk, ha csak meg szeretnénk adni, hogy a gép által alapértelmezett értéket kapják meg majd az osztályobjektumok. Ebben az esetben nem tudunk hibakezelést, sem írni. | osztaly\_neve**() = default;** |

|  |  |
| --- | --- |
| Ha viszont, mindenhova alapértelmezett nulla értéket, akarunk megadni, akkor be tudjuk manuálisan is állítani, úgy hogy tudunk közben hibakezelést is alkalmazni. | osztaly\_neve**()**  **{**  **this -> valtozo\_1 = 0;**  **this -> valtozo\_2 = „”;**  **}** |

### Paraméterezett konstruktor

Abban az esetben, ha paraméterezett konstruktort használunk, akkor amikor létrehozunk egy osztályobjektumot meg kell adni a konstruktor által kért értékeket.

|  |  |
| --- | --- |
| Abban az esetben használjuk, ha nem szeretnénk hibakezelést végrehajtani. | osztaly\_neve**(**int v1, string v2**):** valtozo\_1(v1), valtozo\_2(v2) **{}** |

|  |  |
| --- | --- |
| Abban az esetben használjuk, ha szeretnénk hibakezelést végrehajtani. Pl. Ha v1 = 0, akkor írja ki, hogy a változó értéke nem megfelelő, és alapértelmezett értéket állított be neki a rendszer. | osztaly\_neve**(**int v1, string v2**)**  **{**  **this ->** valtozo\_1 **=** v1**;**  **this ->** valtozo\_2 **=** v2**;**  **}** |

## Lekérdező- és beállítómetódusok

### Lekérdezőmetódus

*A lekérdező metódust abban az esetben használjuk, ha szeretnénk megtudni egy osztályon belüli változó értékét. A lekérdező metódus minden esetben függvény!!!*

|  |  |
| --- | --- |
| Először meg kell adni a változó típusát, majd a metódus nevét, ahhoz hogy az értéke ne tudjon változni, hozzá kell írni a **const** kulcsszót is. Ezután a kapcsos zárójelek között kell megadni azt, hogy melyik értékkel térjen vissza a függvény. ☺ | int **GET**\_valtozo\_1() **const**  **{**  **return this ->** valtozo\_1;  **}** |

### Beállítómetódus

*A beállítómetódussal tudjuk hozzárendelni a bekért értéket az osztály egy változójához. A beállítómetódus minden esetben eljárás!!!*

|  |  |
| --- | --- |
| Először meg kell adni a **void** kulcsszót, majd a metódus nevét, ezután meg kell adni, hogy annak az értéknek mi a típusa amit megszeretnénk adni. Ezután a kapcsos zárójelek között kell megadni azt, hogy melyik változónak legyen az értéke a bekért változó. ☺ | void **SET**\_valtozo\_1(int v1)  **{**  **this ->** valtozo\_1 = v1;  **}** |

#### További információk:

<https://www.w3schools.com/cpp/cpp_encapsulation.asp>

<https://www.geeksforgeeks.org/c-plus-plus/?ref=shm#cpp_oops>

**#pragma once**

**#include <string>**

**#include<iostream>**

**using namespace std;**

**class** Allatok

**{**

**public**:

**Allatok(**int azon, string name, int db, string color**)**

**{**

**this->**azonosito = azon;

**this->**nev = name;

if (db >= 0)

{

**this->**hany\_darab = db;

}

else

{

cout << "\nCsak 0-t, vagy nagyobb számot adjon meg. A db számot 0-ra állítottuk."<<endl;

**this->**hany\_darab = 0;

}

**this->**szin = color;

**}**

**~Allatok()=default;**

**int GET\_azonosito() const {**

**return this->**azonosito;

**}**

**string GET\_nev() const {**

**return this->** nev;

**}**

**int GET\_db() const {**

**return this->**hany\_darab;

**}**

**string GET\_szin() const {**

**return this->**szin;

**}**

**void SET\_azonosito(**int sa**) {**

**this->**azonosito=sa;

**}**

**void SET\_nev(**string sn**) {**

**this->**nev=sn;

**}**

**void SET\_db(**int sd**) {**

**this->**hany\_darab=sd;

**}**

**void SET\_szin(**string ssz**) {**

**this->**szin=ssz;

**}**

**void Allat\_tulajdonsagai() {**

cout << azonosito << " " << nev << " " << hany\_darab << " " << szin;

**}**

**private**:

int azonosito;

string nev;

int hany\_darab;

string szin;

**};**

# Öröklődés

*Amikor egy osztály* ***tulajdonságokat és jellemzőket származtat egy másik osztályból****, azt öröklődésnek nevezzük. Az öröklődés egy olyan funkció vagy folyamat, amelyben új osztályok jönnek létre a meglévő osztályokból.*

***Alosztálynak*** *nevezzük, azt az osztályt, amely egy másik osztály tulajdonságait örökli.* ***Szuperorsztálynak*** *pedig, azt az osztályt amelynek tulajdonságait örökli egy alosztály.*

## Szuperosztály

|  |  |
| --- | --- |
| Ha azt szeretnénk, hogy egy osztályt tudjunk szuperosztályként kezelni, akkor a változóit nem **privát-**ként kell megadni, hanem **protected**-ként.  A **protected** hozzáférésnél, a védettnek deklarált osztályváltozók nem lesznek elérhetők az osztályon kívülm de az adott osztály bármely alosztálya elérheti őket. | **class** osztaly\_nev  {  **protected**:  string valtozo\_1;  **public**:  osztaly\_nev ()=default;  ~ osztaly\_nev ()=default;  }; |

## Alosztály

|  |  |
| --- | --- |
| Az alosztály létrehozásakor a : teszi lehetővé, hogy elérje a szülő osztály számára publikus változóit. Amikor írjuk az osztály számára a konstruktor, akkor is be kell írni a szülő osztály összes konstruktorának megfelelő elemét, ha az alosztály nem igényli azt!!! ☹ | **class** osztaly\_nev : **public** szulo  {  **public**:  osztaly\_nev (int v1, int v2) : szulo (v1, v2) {}  ~ osztaly\_nev ()=default;  }; |

## Main hivatkozás

|  |  |
| --- | --- |
| Ahhoz, hogy hivatkozni tudjunk az alosztály által egy szülő osztálybeli függvényre, meg kell adni az alosztály nevét és azt, hogy mi legyen az objektum neve ezáltal tudjuk deklarálni. Amikor a nevet és azt függvényt adjuk meg amire hivatkozni szeretnénk, akkor fogjuk meghívni a függvényt. | **int main**  {  Alosztaly a1(5, 6);  a1.szulo\_fv();  } |

#### További információk:

<https://www.geeksforgeeks.org/cpp-inheritance-access/?ref=next_article>

<https://mogi.bme.hu/TAMOP/c++programozas/ch03.html>

## Szuperosztály

**#pragma once**

**#include <string>**

**#include <iostream>**

**using namespace std;**

**class** Termekek

{

**protected**://közös tulajdonságok

string nev;

int ar;

int afa;

string tarolas;

**public**:

Termekek(string n, int a, int af, string ta) {

this->nev = n;

this->ar = a;

this->afa = af;

this->tarolas = ta;

}

~Termekek() = default;

string getNev() const {

return this->nev;

}

int getAr() const {

return this->ar;

}

int getAfa() const {

return this->afa;

}

string getTarolas() const {

return this->tarolas;

}

void setNev(string sn) {

this->nev = sn;

}

void setAr(int sa) {

this->ar=sa;

}

void setAfa(int saf) {

this->afa=saf;

}

void setTarolas(string st) {

this->tarolas=st;

}

};

## Alosztály

**#pragma once**

**#include <string>**

**#include <iostream>**

**using namespace std;**

**class** Hus\_felek:**public** Termekek

{

**public**:

Hus\_felek(string n, int a, int af, string t, string em, string f) : Termekek(n, a, af,t) {

this->elkeszitesi\_mod = em;

this->fajta = f;

}

~Hus\_felek() = default;

string GetElk\_mod() const {

return this->elkeszitesi\_mod;

}

string GetFajta() const {

return this->fajta;

}

void SetElk\_mod(string sem) {

this->elkeszitesi\_mod=sem;

}

void SetFajta(string sf){

this->fajta=sf;

}

void osszadat() {

cout << nev << " " << ar << " " << afa << " " << tarolas << " " << elkeszitesi\_mod << " " << fajta;

}

**private**:

string elkeszitesi\_mod;

string fajta;

};

## Main

**#include "Termekek.h"**

**#include "Hus\_felek.h"**

**#include "Tejtermek.h"**

**#include "Gyumolcs\_Zoldseg.h"**

**#include <string>**

**#include <iostream>**

**using namespace std;**

int main()

{

Hus\_felek h1("Lapocka", 2000, 5, "fagyasztó", "sütés", "disznó");

h1.osszadat();

Tejtermek t1("Tej", 300, 27, "Hűtő (0-5C)", "Laktózmentes", false);

t1.osszadat();

}

# Stringek

*A stringek olyan objektumok, amelyek karaktersorozatokat képviselnek. A szabványos karakterláncosztály támogatja az ilyen objektumokat.*

## Elemek lekérdezése

|  |  |  |
| --- | --- | --- |
| Működése | Példa program | Kimenet |
| Üres string létrehozása | string s = „”; |  |
| Adott indexű elem lekérdezéséhez használjuk | string s = „sárgarépa”;  cout << **s[4];** | g |
| string s = „sárgarépa”;  cout << **s.at(4);** | g |
| A string utolsó elemét kicseréli adott karakterre. | string s = „sárgarépa”;  **s.back()=’i’;**  cout << s; | sárgarépi |
| A string első elemét kicseréli adott karakterre. | string s = „sárgarépa”;  **s.front()=’S’;**  cout << s; | Sárgarépa |

## Módosítók

|  |  |  |
| --- | --- | --- |
| Stringek összefűzése | string s = „sárga”;  string r = „répa”;  string s **+=** sr;  cout << s; | sárgarépa |
| Karakterlánc bővítése további karakterlánc hozzáadásával. | string str=””; string s=”sárga”;  **str.append(s);**  cout << str; | sárga |
| string str=””; string s=”sárga”;  **str.append(s,1,3);**  cout << str; | árg |
| string str=””; string s=”sárga”;  **str.append(s,3);**  cout << str; | sár |
| string str=””;  str.append(5,’.’);  cout << str; | ….. |
| string str=””; string s=”sárga”;  str.append(s.begin()+2, str.end());  cout << str; | rga |
| Bővíti a string hosszát eggyel. | string str=””;  str.push\_back(’s’);  cout << str; | s |
| Új értéket rendel a stringhez lecserélve a tartalmát | string str=””; string s=”sárgarépa”;  **str.assign(s);**  cout << str; | sárgarépa |
| string str=””; string s=”sárgarépa”;  **str.assign(s,3,6);**  cout << str; | garépa |
| string str=””; string s=”sárgarépa”;  **str.assign(s,3);**  cout << str; | sár |
| string str=””; string s=”sárgarépa”;  **str.assign(5,’\*’);**  cout << str; | \*\*\*\*\* |
| Karakterláncot szúr be adott p pont után | string str=”ez itt kérdés”, t=”a ”, r=”lenni vagy nem lenni”  **str.insert(**t, 7**);**  cout << str; | ez itt a kérdés |
| Töröl egy adott részt a stringből | str str = „Lenni vagy nem lenni, ez itt a kérdés”;  str.erase(10,8);//start: 10, count: 8  cout << str; | Lenni vagyni, ez itt a kérdés |
| str str = „Lenni vagy nem lenni, ez itt a kérdés”;  str.erase(str.begin()+8);//8. karaktert törli  cout << str; | Lenni vay nem lenni, ez itt a kérdés |
| Akkor használjuk ha az egész karakterláncban szeretnénk keresni az adott szöveget. | std::size\_t position;  position = s.find("el", 0);  if (position == std::string::npos)  std::cout << "nem talalhato\n"; | Ha npos-t ad vissza akkor nem található |
| Ha adott helyen keressük az egyezést akkor használjuk.  (-1: kisebb  0: egyezik a két string  1: nagyobb) | s = "pelda";  s.compare("pelda");  s.compare(3, 2, "ld"); | Megnézi, hogy szerepel-e benne karakterkód alapján. |

#### További információk:

<https://cplusplus.com/reference/string/string/>

# Fájlkezelés

*C++-ban az* ***fstream*** *könyvtá teszi lehetővé számunkra, hogy fájlokkal dolgozzunk. A könyvtár használatához be kell írni a fejlécbe az <iostream>, és <fstream>.*

## Fájl létrehozása és írása

|  |  |
| --- | --- |
| Ahhoz hogy létretudjunk hozni egy fájlt az ofstream parancsot kell használni. Amikor létrehozzuk az objektumot, akkor meg kell adni argumentumként a fájl nevét. Fontos hogy minden műveletet be kell zárni a **Close** parancs segítségével. | **#include <iostream>**  **#include <fstream>**  **#include <string>**  **using namespace std;**  int main() {  **ofstream** fajl("fajl\_nev.txt");  fajl **<<** „Amit bele szeretnénk írni";  **fajl.close();**  } |

## Fájl beolvasás

|  |  |
| --- | --- |
| Mielőtt megkezdjük a fájlbeolvasást létre kell hozni egy stringet amibe beolvassuk majd a sorokat. Amikor megnyitjuk az **ifstream**-mal a beolvasást, akkor meg kell adni argumentumként a beolvasandó file nevét, és a konstruktorát. Konstruktorok  * **in:** az olvasásra megnyitott fájl támogatja a beviteli műveleteket * **out:** A fájl írásra van megnyitva -> támogatja a kimeneti műveleteket * **binary:** A műveleteket szöveg helyett bináris módban hajtják végre * **ate:** A kimeneti pozíció a fájl végén kezdődik * **app:** Minden kimeneti művelet a fájl végén történik, hozzáfűzi az adatot, nem törli. * **trunc:** Minden olyan tartalom, amely a fájlban a megnyitás előtt létezett eldobásra kerül. | **#include <iostream>**  **#include <fstream>**  **#include <string>**  **using namespace std;**  int main()  {  string beolvasott\_sor;  **ifstream** beolvasas("fajl\_nev.txt", **fstream**:construct);  **while(getline(**beolvasas, beolvasott\_sor**))**  **{**  cout<<beolvasott\_sor;  **}**  beolvasas**.close();**  } |

Ha megszeretnénk tudni, hogy a fájl nyitva van-e akkor az **is\_open()** parancsot kell használni. Ennek a visszatérési értéke egy **bool**-érték.

Ha beszeretnénk zárni egy fájlt, akkor a **close-et** kell használni.

|  |  |
| --- | --- |
| A std::istream::gcount a C++ Standard Library egy tagfüggvénye, amely a legutóbbi olvasási művelet során ténylegesen olvasott karakterek számát adja vissza. | char szoveg[meret];  cin.getline(szoveg, meret);  cout << cin.gcount(); |

#### További információk:

<https://cplusplus.com/reference/fstream/ifstream/>

## Bináris fájlkezelés

*A bináris fájl olyan fájl, amely nyers formátumban tárolja az adatokat a memóriában tárolt módon. Például a számok binárisan tárolódnak a memóriában.*

*A bináris fájlok formázatlan I/O-t hajtanak végre nyers adatokkal. Így a bináris formátum és az ASCII szövegformátum közötti átalakítás bináris fájlok esetén nem történik meg.*

*A bináris fájl megnyitásához meg kell adni az ios::binary nyílt módot. A bináris bemeneti/kimeneti műveletek read() és write() metódusokkal hajthatók végre.*

***#include <fstream>***

***#include <iostream>***

***#include <string>***

***using namespace std;***

***int main()***

***{***

***setlocale(LC\_ALL, "hun");***

*int n = 5;*

***ifstream*** *input;*

***ofstream*** *output;*

***char buffer[256];***

*output.****open****("pelda.bin",* ***std::ios::out | std::ios::binary);***

*output.****write****((char\*)&n, sizeof(int));*

*output.****write****(buffer, 256);*

*//output.write((char\*)&M, sizeof(Mercedes));*

*output.****close****();*

*input.****open****("pelda.bin",* ***std::ios::in | std::ios::binary);***

*input.****read****((char\*)&n, sizeof(int));*

*input.****read****(buffer, 256);*

*input.****close****();*

***return 0;***

*}*

## Sima fájlkezelés

**#include <fstream>**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**int main()**

{

setlocale(LC\_ALL, "hun");

//Fájl létrehozása

**ofstream** Korte("korte.txt");

Korte **<<** "A körte egy gyümölcs";

Korte.**close**();

//Fájl beolvasás

string beolvasott\_sor;

**ifstream** beolvasas("korte.txt", fstream::app);

while (**getline**(beolvasas,beolvasott\_sor))

{

**cout** << beolvasott\_sor << endl;

}

beolvasas.**close**();

//Karakterek száma

char szoveg[256];

cout << "Adjon meg egy szót: ";

cin.**getline**(szoveg,256);

cout << cin.**gcount**() << " a beolvasott karakterek száma.";

}