Arduino osztály és library használata

Ezt egy tutorialnak szántam azok számára, akiknek már van programozási gyakorlatuk, de még nem használták a C++ nyelv osztályait az Arduinon, de ki akarják próbálni.

Arra a "rendkívül bonyolult" :-) feladatra akarunk programot írni, hogy kiírja,
hogy melyik autóhoz mennyi festék kell. A festék mennyisége legyen a felület * 1.15

Akkor ESP32-n ez a program kiszámolja, és kiírja.
Természetesen a kiírásban használhatnánk a Serial. print-et is.
A Serial.begin(115200) mindenképp szükséges, mert mind a cout, mind a
Serial.print a soros monitorra ír ki. Ezért a program feltöltése előtt nyissuk meg
a soros monitort!
A program:

```
#include <iostream>
using namespace std;

string tulajdonos ;
string szine ;
float felulete ;
float mennyiseg ;
void setup() {

  Serial.begin(115200);
  delay(2000);

  tulajdonos = "Pista" ;
  szine = "piros" ;
  felulete = 1.4 ;
  mennyiseg = felulete * 1.15 ;

  cout << tulajdonos << " autoja"<< "\n"  ;
  cout << "auto szine: " << szine << "\n"  ;
  cout << "ennyi festek kell: " << mennyiseg  << "\n\n" ;


  tulajdonos = "Feri" ;
  szine = "randaMatt" ;
  felulete = 0.8 ;
  mennyiseg = felulete * 1.15;

  cout << tulajdonos << " autoja"<< "\n"  ;
  cout << "auto szine: " << szine << "\n"  ;
  cout << "ennyi festek kell: " << mennyiseg << "\n\n" ;

} ;

void loop() {
} ;
```

Ezt írja ki soros monitorra:

Pista autoja
auto szine: piros
ennyi festek kell: 1.61

Feri autoja
auto szine: randaMatt
ennyi festek kell: 0.92

****

Függvény

Mivel több autóra akarjuk kiszámítani, ezért készítünk egy függvényt.

```
#include <iostream>
using namespace std;

void kiszamol (string tulajdonos, string szine, double felulete) {
  double mennyiseg ;
  mennyiseg = felulete * 1.15 ;
  cout << tulajdonos << " autoja"<< "\n"  ;
  cout << "auto szine: " << szine << "\n"  ;
  cout << "ennyi festek kell: " << mennyiseg << "\n\n" ;
};

void setup() {

// ez kell, hogy a cout a soros monitorra irjon
  Serial.begin(115200);
  delay(2000);

 kiszamol ("Pista","piros", 1.4) ;
 kiszamol ("Feri","randaMatt", 0.8) ;

} ;

void loop() {

} ;
```

Ezt írja ki a soros monitorra:

Pista autoja
auto szine: piros
ennyi festek kell: 1.61

Feri autoja
auto szine: randaMatt
ennyi festek kell: 0.92

****

Struktúra

Készíthetünk egy struktúrát, majd minden autóhoz létrehozunk egy példányt, és abban tárolhatjuk az egyes autók adatait:

```
#include <iostream>
using namespace std;

struct autoStruct{
  string tulajdonos ;
  string szine ;
  double felulete ;
  double mennyiseg ;
};

autoStruct pistaAutoja;
autoStruct feriAutoja;

void setup() {

  Serial.begin(115200);
  delay(2000);

  pistaAutoja.tulajdonos = "Pista" ;
  pistaAutoja.szine = "piros" ;
  pistaAutoja.felulete = 1.4 ;
  pistaAutoja.mennyiseg = pistaAutoja.felulete * 1.15 ;

  cout << pistaAutoja.tulajdonos << " autoja"<< "\n"  ;
  cout << "auto szine: " << pistaAutoja.szine << "\n"  ;
  cout << "ennyi festek kell: " << pistaAutoja.mennyiseg  << "\n\n" ;


  feriAutoja.tulajdonos = "Feri" ;
  feriAutoja.szine = "randaMatt" ;
  feriAutoja.felulete = 0.8 ;
  feriAutoja.mennyiseg = feriAutoja.felulete * 1.15;

  cout << feriAutoja.tulajdonos << " autoja"<< "\n"  ;
  cout << "auto szine: " << feriAutoja.szine << "\n"  ;
  cout << "ennyi festek kell: " << feriAutoja.mennyiseg << "\n\n" ;

} ;

void loop() {
} ;
```

Ezt írja ki:

Pista autoja
auto szine: piros
ennyi festek kell: 1.61

Feri autoja
auto szine: randaMatt
ennyi festek kell: 0.92

****

autoOsztaly publickal

Ha tovább akarunk lépni, hogy megismerjük az osztályokat, akkor a következő programot nézzük meg. Már most előrebocsátom, hogy illetlen, pancser módszernek szokás tekinteni, ha az osztály változóit publicként adjuk meg, azaz a program bármely részéből írni olvasni lehet, de a következő programban már majd korrigáljuk.

Deklarálunk egy autoOsztaly osztályt, majd a példányosítás során létrehozunk mindkét autóra egy-egy példányt. Ezekben minden autónak van saját színe, stb., nem zavar, hogy a másik autónak más a színe, mert minden egyes példány külön letárolja a saját változóit.
Az osztály példányait a konstruktorával hozzuk létre. A konstructor neve megegyezik az osztály nevével. Lehet egy osztálynak több konstruktora is, ha a paramétereik száma és/vagy típusa különbözik (többalakúság). A fordító ez alapján tudja, melyikkel kell létrehozni az adott példányt. Itt most két konstruktort adtam meg, és az egyik példányt (pistaAutoja) az egyparaméteres, a másikat (feriAutoja) a kétparaméteres konstruktorral hoztam létre. Megemlítem, hogy van destruktor is, de itt arra nem térek ki.

```
#include <iostream>
using namespace std;

class autoOsztaly {
  private : string tulajdonos ;
  private : string szine ;
  private : double felulete ;

  public : autoOsztaly(string tulaj){
    tulajdonos = tulaj ;
  };

  public : autoOsztaly(string tulaj, string szin){  //  kétparameteres konstruktor
    tulajdonos = tulaj ;
    szine = szin ;
```

```
  } ;

  public : void setSzine(string autoSzine){
    szine  = autoSzine ;
  };

  public : string getSzine(){
    return  szine ;
  };

  public : string getTulaj(){
    return tulajdonos ;
  };

  public : void setFelulete(double felulet){
    felulete = felulet ;
  };

  public : double mennyiseg() {
    return   felulete * 1.15 ;
  } ;

} ;  // autoOsztaly vege


  autoOsztaly pistaAutoja("Pista");
  autoOsztaly feriAutoja("Feri","randaMatt");

void setup() {

  Serial.begin(115200);
  delay(2000);

  pistaAutoja.setSzine("piros") ;
  pistaAutoja.setFelulete(1.4) ;

  feriAutoja.setFelulete(0.8) ;

  cout  << pistaAutoja.getTulaj() << " autoja " << "\n"  ;
  cout  << "auto szine: " << pistaAutoja.getSzine()<< "\n"  ;
  cout  << "ennyi festek kell: " << pistaAutoja.mennyiseg() << "\n\n"  ;

  cout  << feriAutoja.getTulaj() << " autoja " << "\n"  ;
  cout  << "auto szine: " << feriAutoja.getSzine()<< "\n"  ;
  cout  << "ennyi festek kell: " << feriAutoja.mennyiseg() << "\n\n"  ;

} ; // end setup

void loop() {

} ;
```

Kiírás:

Pista autoja
auto szine: piros

ennyi festek kell: 1.61

Feri autoja
auto szine: randaMatt
ennyi festek kell: 0.92

****

autoOsztaly private-tal

Már említettem, hogy nem szokás a változókat publicként deklarálni, hanem az osztálynak külön tagfüggvényeket írunk a változók értékadására, és olvasására. Ekkor a változókat private-ként adjuk meg, ezzel a program más részein közvetlenül nem érhetők el, a példányokban egységbe zártuk.
Erre nézzük a következő példát.

```
#include <iostream>
using namespace std;

class autoOsztaly {
  private : string tulajdonos ;
  private : string szine ;
  private : double felulete ;

  public : autoOsztaly(string tulaj){
    tulajdonos = tulaj ;
  };

  public : autoOsztaly(string tulaj, string szin){  //  kétparameteres konstruktor
    tulajdonos = tulaj ;
    szine = szin ;
  } ;

  public : void setSzine(string autoSzine){
    szine  = autoSzine ;
  };

  public : string getSzine(){
    return  szine ;
  };

  public : string getTulaj(){
    return tulajdonos ;
  };

  public : void setFelulete(double felulet){
    felulete = felulet ;
  };

  public : double mennyiseg() {
    return   felulete * 1.15 ;
  } ;
```

```
} ;  // autoOsztaly vege


  autoOsztaly pistaAutoja("Pista");
  autoOsztaly feriAutoja("Feri","randaMatt");

void setup() {

  Serial.begin(115200);
  delay(2000);

  pistaAutoja.setSzine("piros") ;
  pistaAutoja.setFelulete(1.4) ;

  feriAutoja.setFelulete(0.8) ;

  cout  << pistaAutoja.getTulaj() << " autoja " << "\n"  ;
  cout  << "auto szine: " << pistaAutoja.getSzine()<< "\n"  ;
  cout  << "ennyi festek kell: " << pistaAutoja.mennyiseg() << "\n\n"  ;

  cout  << feriAutoja.getTulaj() << " autoja " << "\n"  ;
  cout  << "auto szine: " << feriAutoja.getSzine()<< "\n"  ;
  cout  << "ennyi festek kell: " << feriAutoja.mennyiseg() << "\n\n"  ;

} ; // end setup

void loop() {

} ;
```

Kiírás:

Pista autoja
auto szine: piros
ennyi festek kell: 1.61

Feri autoja
auto szine: randaMatt
ennyi festek kell: 0.92

****

Prototípussal

Ha nem akarjuk, hogy a programunk elején ott legyen az osztály hosszú leírása, akkor megtehetjük, hogy csak a prototípusát adjuk meg a program elején, és hogy melyik tagfüggvény mit csinál, azt a program végén adjuk meg. Hogy ennek mi az értelme? Itt most az, hogy onnan már csak egy lépés a könyvtárak használata.

```
#include <iostream>
using namespace std;

class autoOsztaly {  // prototipus megadasa

  string tulajdonos ;
  string szine ;
  double felulete ;
  public : autoOsztaly(string tulaj) ;
  public : autoOsztaly(string tulaj, string szin) ;
  public : void setSzine(string autoSzine) ;
  public : string getSzine() ;
  public : string getTulaj() ;
  public : void setFelulete(double felulet) ;
  public : double mennyiseg() ;

} ;  // autoOsztaly vege


  autoOsztaly pistaAutoja("Pista");
  autoOsztaly feriAutoja("Feri","randaMatt");

void setup() {

  Serial.begin(115200);
  delay(2000);

  pistaAutoja.setSzine("piros") ;
  pistaAutoja.setFelulete(1.4) ;

  feriAutoja.setFelulete(0.8) ;

  cout  << pistaAutoja.getTulaj() << " autoja " << "\n"  ;
  cout  << "auto szine: " << pistaAutoja.getSzine()<< "\n"  ;
  cout  << "ennyi festek kell: " << pistaAutoja.mennyiseg() << "\n\n"  ;

  cout  << feriAutoja.getTulaj() << " autoja " << "\n"  ;
  cout  << "auto szine: " << feriAutoja.getSzine()<< "\n"  ;
  cout  << "ennyi festek kell: " << feriAutoja.mennyiseg() << "\n\n"  ;

} ; // end setup

void loop() {

} ;


// itt adjuk meg tenylegesen, hogy melyik tagfuggveny mit csinal

  autoOsztaly:: autoOsztaly(string tulaj){
    tulajdonos = tulaj ;
  };

  autoOsztaly:: autoOsztaly(string tulaj, string szin){
```

```
    tulajdonos = tulaj ;
    szine = szin ;
  } ;

  void autoOsztaly::  setSzine (string autoSzine) {
    szine  = autoSzine ;
  };

  string autoOsztaly::  getSzine(){
    return  szine ;
  };

  string autoOsztaly:: getTulaj(){
    return tulajdonos ;
  };

  void autoOsztaly::  setFelulete(double felulet){
    felulete = felulet ;
  };

  double autoOsztaly::  mennyiseg() {
    return   felulete * 1.15 ;
  } ;
```

****

Az előbb már említettem, hogy innen már csak egy lépés a könyvtárazás, azaz a library használata.
Íme:

Library

A program elejéről a prototípus megadását másoljuk ki egy fájlba.

Legyen ez az

autoOsztaly.h

```
#include <iostream>
using namespace std;

class autoOsztaly {  // prototipus megadasa

  string tulajdonos ;
  string szine ;
  double felulete ;
  public : autoOsztaly(string tulaj) ;
  public : autoOsztaly(string tulaj, string szin) ;
```

```
  public : void setSzine(string autoSzine) ;
  public : string getSzine() ;
  public : string getTulaj() ;
  public : void setFelulete(double felulet) ;
  public : double mennyiseg() ;

} ;  // autoOsztaly vege
```

A program végén lévő részt pedig egy .cpp fájlba úgy, hogy az elejére írjuk be, az #include "autoOsztaly.h" sort!

autoOsztaly.cpp

```
#include "autoOsztaly.h"
// itt adjuk meg tenylegesen, hogy melyik tagfuggveny mit csinal

  autoOsztaly:: autoOsztaly(string tulaj){
    tulajdonos = tulaj ;
  };

  autoOsztaly:: autoOsztaly(string tulaj, string szin){
    tulajdonos = tulaj ;
    szine = szin ;
  } ;

  void autoOsztaly::  setSzine (string autoSzine) {
    szine  = autoSzine ;
  };

  string autoOsztaly::  getSzine(){
    return  szine ;
  };

  string autoOsztaly:: getTulaj(){
    return tulajdonos ;
  };

  void autoOsztaly::  setFelulete(double felulet){
    felulete = felulet ;
  };

  double autoOsztaly::  mennyiseg() {
    return   felulete * 1.15 ;
  } ;
```

A két fájlt mentsük abba a könyvtárba, ahol a .ino fájlunk is van

A főprogramból töröljük a kimásolt részeket, és írjuk be az elejére, hogy #include "autoOsztaly.h"

a .ino programunk ez lesz:

```
#include <iostream>
#include "autoOsztaly.h"

using namespace std;

  autoOsztaly pistaAutoja("Pista");
  autoOsztaly feriAutoja("Feri","randaMatt");

void setup() {

  Serial.begin(115200);
  delay(2000);

  pistaAutoja.setSzine("piros") ;
  pistaAutoja.setFelulete(1.4) ;

  feriAutoja.setFelulete(0.8) ;

  cout  << pistaAutoja.getTulaj() << " autoja " << "\n"  ;
  cout  << "auto szine: " << pistaAutoja.getSzine()<< "\n"  ;
  cout  << "ennyi festek kell: " << pistaAutoja.mennyiseg() << "\n\n"  ;

  cout  << feriAutoja.getTulaj() << " autoja " << "\n"  ;
  cout  << "auto szine: " << feriAutoja.getSzine()<< "\n"  ;
  cout  << "ennyi festek kell: " << feriAutoja.mennyiseg() << "\n\n"  ;

} ; // end setup

void loop() {

} ;
```

Futtathatjuk a programot, és ugyanazt az eredményt fogja adni, mint eddig. Természetesen ez csak egy nagyon rövid kivonat, de ha ezeket feltöltöd és kipróbálod, akkor már a netről hozzá szedett információkat könnyebben megérted.