L01: kedd 10-12 rendelkezésre álló idő:15 **perc**

Programozás alapja 2.	1. beszámoló dolgozat. Kurz/**Terem**: L01/	2020.03.03.	Elért pontszám:
Név:		**Neptun:**	

Az első feladatnál minden bejelölt válasz 1 pont, ha helyes, -1 pont, ha hibás! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív.
Labor e.d. (beugró): 1. feladatból min 2 pont.

1. Feladat (4p)

Jelölje, hogy mely állítás(ok) **hamis(ak)** a C++ nyelvre!

- ☐ Referencia típusú változó átadható paraméterként.
- ☐ Minden programot `using namespace std;` direktívával kell kezdeni.
- ☐ A **new** képes kivételt generálni.
- ☐ A névterek egymásba ágyazhatók.
- ☐ Függvényprototípus használata kötelező, ha a függvény a használat előtt nincs definiálva.
- ☐ A **throw** utasítás végtelen ciklusból való kilépésre való.
- ☐ Kivételt nem csak abban a fájlban lehet elfogni, ahol a **throw** utasítás van.
- ☐ Minden programot **#include <iostream>** direktívával kell kezdeni.

Jelölje, hogy mely állítás(ok) **igaz(ak)** az alábbi C++ kódrészletre!

```
{ int i; while (std::cin >> i); }
```

- ☐ Addig olvas be egész számokat, amíg az inputon egész formátumnak megfelelő adatok érkeznek
- ☐ A kódrészlet hibás, mert az **i** változó nem kap értéket.
- ☐ Végtelen ciklus.

2. **Tervezzen** egy olyan osztályt (**Ido**), ami idő (óra, perc) tárolására alkalmas! Az osztálynak (6p)
   - legyen paraméter nélkül hívható konstruktora, ami 0 óra 0 percet állít be;
   - legyen olyan konstruktora is, amivel az óra és a perc is megadható;
   - legyen olyan operátora (+), amivel egy időponthoz egész (int) órát lehet adni jobbról! Az eredmény az összeadás műveletnél megszokott módon új objektumban keletkezzen! Amennyiben az összeadás során keletkező objektumban az óra értéke meghaladná a 24 órát, úgy dobjon **const char*** típusú kivételt!

   Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni.
   Működjön helyesen az alábbi kódrészlet:

```
Ido t0, t10(10, 0), t23(23, 05);// 00:00, 10:00, 23:05 időpontokat hoz létre
t0 = t10 + 1;                   // t0-ba 11:00 kerül, t10 változatlan marad
t23 + 2;                        // kivétel keletkezik
```

```
class Ido {
    int ora;
    int perc;
public:
    Ido(int o = 0, int p = 0) :ora(o), perc(p) {}
    Ido operator+(int rhs) const;
};
Ido Ido::operator+(int rhs) const {
    if (ora + rhs > 24) throw "Ora > 24";
    return Ido(ora + rhs, perc);
}
```

L02: csütörtök 10-12 rendelkezésre álló idő:15 **perc**

Programozás alapja 2.	1. beszámoló dolgozat. Kurz/**Terem**: L02/	2020.03.05.	Elért pontszám:
Név:		**Neptun:**	

Az első feladatnál minden bejelölt válasz 1 pont, ha helyes, -1 pont, ha hibás! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív.
Labor e.d. (beugró): 1. feladatból min 2 pont.

1. Feladat (4p)

Jelölje, hogy mely állítás(ok) **hamis(ak)** a C++ nyelvre!

- ☐ A struktúra egy osztály.
- ☐ A **new** sosem generál kivételt.
- ☐ Az értékadó operátor nem terhelhető túl.
- ☐ A **delete[]** egy operátor.
- ☐ A C nyelvben ismert realloc függvényt a **renew** operátor helyettesíti.
- ☐ A névterek egymásba ágyazhatók.
- ☐ Referencia típusú változó átadható paraméterként.
- ☐ Egy változót többször is lehet deklarálni, de definiálni csak egyszer lehet.

Jelölje, hogy mely állítás(ok) **igaz(ak)** az alábbi C++ kódrészletre!

```
void swap(int& a, int& b) {
    int c = a;
    a = b;
    b = c; }
```

- ☐ A függvény felcseréli a referencia paraméterként kapott két változó adatát.
- ☐ A függvénynek nincs mellékhatása..
- ☐ A függvény hibás, mert lokális változónak referenciát ad értékül.

3. **Tervezzen** egy olyan osztályt (**Honap**), ami hónap sorszámának tárolására alkalmas! Az osztálynak (6p)
   - legyen paraméter nélkül hívható konstruktora, ami 3. hónapot (márciust) állít be;
   - legyen olyan konstruktora is, amivel megadható a hónap;
   - legyen olyan operátora (+), amivel egy Honaphoz egész (int) hónapot lehet adni jobbról! Az eredmény az összeadás műveletnél megszokott módon új objektumban keletkezzen! Amennyiben az összeadás során keletkező objektumban a hónap értéke meghaladná a 12-t, úgy dobjon **const char*** kivételt!

   Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni.
   Működjön helyesen az alábbi kódrészlet:

```
Honap h3, h10(10);          // 3. és 10. hónapot hoz létre
h3 = h10 + 1;               // h3-ba 11. hónap, kerül, h10 változatlan
h3 + 2;                     // kivétel keletkezik
```

```
class Honap {
    int ho;
public:
    Ho(int h = 3) :ho(h) {}
    Ho operator+(int rhs) const;
};
Ho Ho::operator+(int rhs) const {
    if (ho + rhs > 12) throw "Ho > 12";
    return Ho(ho + rhs);
}
```

L03: Csütörtök 15-17 rendelkezésre álló idő:15 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2020.03.05. Kurz/**Terem**: L03/	Elért pontszám:
Név: **Neptun:**	

Az első feladatnál minden bejelölt válasz 1 pont, ha helyes, -1 pont, ha hibás! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív.
Labor e.d. (beugró): 1. feladatból min 2 pont.

1. Feladat (4p)

Jelölje, hogy mely állítás(ok) **hamis(ak)** a C++ nyelvre!

- ☐ **`inline`** függvénynek lehet default paramétere.
- ☐ **realloc** helyett **renew** utasítást kell használni.
- ☐ Destruktornak csak konstans paramétere lehet.
- ☐ A referencia egy alternatív név.
- ☐ Függvénysablonnak nem lehet paramétert átadni.
- ☐ A ciklus feltételében is lehet változót deklarálni. Pl: **`while (int i = f())`**…
- ☐ Konstans tagfüggvény nem változtathatja meg az objektum állapotát.
- ☐ Egy változót többször is lehet deklarálni.

Jelölje, hogy mely állítás(ok) **igaz(ak)** az alábbi C++ kódrészletre!

```
{ struct S { int a; }; S so; }
```

- ☐ **`S`** egy osztály.
- ☐ **`S`** egy objektum.
- ☐ **`S`** minden adattagja privát
- ☐ **`so`** destruktora sosem hívódik meg, mert nincs.
- ☐ **`so`** konstruktora sosem hívódik meg, mert nincs.

2. Tervezzen egy osztályt (Tort) törtek tárolására! Az osztály a törtet két egész számként (számláló, nevező) tárolja! Az osztálynak (6p)

- legyen paraméter nélkül hívható konstruktora, ami a számlálót 0-ra, a nevezőt 1-re állítja;
- legyen olyan konstruktora is, amivel a számláló és a nevező is beállítható;
- legyen olyan operátora (/), amivel egy ilyen tört egy egész számmal eloszthető! Az eredmény az osztás műveletnél megszokott módon új objektumban keletkezzen! Nullával való osztás esetén dobjon **const char*** kivételt!

Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni.
Működjön helyesen az alábbi kódrészlet:

```
Tort t0, t1(1,1), t05(1, 2);          // 0/1, 1/1, 1/2
t0 = t1 / 2;                          // t0-ba 1/2 kerül, t1 változatlan marad
t05 = t05 / 0;                        // kivétel keletkezik
```

```
class Tort {
    int szamlalo;
    int nevezo;
public:
    Tort(int szamlalo=0, int nevezo=1) :szamlalo(szamlalo), nevezo(nevezo) {}
    Tort operator/(int) const;
};
Tort Tort::operator/(int n) const {
    if (n == 0) throw "Div 0";
    return Tort(szamlalo, n*nevezo);
}
```

L04: péntek 10-12 rendelkezésre álló idő:15 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2020.03.06. Kurz/**Terem**: L04/	Elért pontszám:
Név: **Neptun:**	

Az első feladatnál minden bejelölt válasz 1 pont, ha helyes, -1 pont, ha hibás! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív.
Labor e.d. (beugró): 1. feladatból min 2 pont.

1. Feladat (4p)

Jelölje, hogy mely állítás(ok) **hamis**(ak) a C++ nyelvre!

- ☐ A **`private`** kulcsszó struktúrában nem szerepelhet.
- ☐ A **`new[]`** képes kivételt generálni.
- ☐ **`inline`** függvényben nem lehet ciklus.
- ☐ A **`delete`** egy operátor.
- ☐ A konstruktor az objektum létrehozásáért felelős.
- ☐ A referenciát kötelező inicializálni, ezért függvényparaméterként nem adható meg
- ☐ A **`new[]`** operátorral foglalt memóriaterületet **`delete[]`** operátorral kell felszabadítani.
- ☐ A névterek egymásba ágyazhatók.

Jelölje, hogy mely állítás(ok) **igaz(ak)** az alábbi C++ kódrészletre!

```
{ class C { int x; }; C co;}
```

- ☐ **`C`** egy objektum.
- ☐ **`C`** egy osztály.
- ☐ **`C`** minden adattagja publikus.
- ☐ C konstruktora nem hívódik meg, mert nincs
- ☐ A kódrészletben memóriaszivárgás lép fel.

2. Tervezzen egy olyan osztályt (Vektor), ami síkbeli pontok helyvektorának tárolására használható! A helyvektort X,Y koordinátákkal tároljuk. Az osztálynak (6p)

- legyen paraméter nélkül hívható konstruktora, ami mindkét koordinátát nullára állítja;
- legyen olyan konstruktora is, amivel mind az X, mind az Y koordináta beállítható;
- legyen olyan operátora (/), amivel egy vektort el lehet osztani egy egész számmal! Az osztás során mind az X, min az Y koordináta osztódik. Az eredmény az osztás műveletnél megszokott módon új objektumban keletkezzen! Nullával való osztás esetén dobjon **const char*** kivételt!

Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni. **Működjön** helyesen az alábbi kódrészlet:

```
Vektor v0, v1(2), v15(1, 5);        // (0, 0); (2,0); (1,5)
v0 = v1 / 2;                        // v0-ba (1,0) kerül, v1 változatlan marad
v15 = v15 / 0;                      // kivétel keletkezik
```

```
class Vektor {
    double x;
    double y;
public:
    Vektor(double x = 0, double y = 0) :x(x), y(y) {}
    Vektor operator/(int rhs) const;
};
Vektor Vektor::operator/(int rhs) const {
    if (rhs == 0) throw "div 0";
    return Vektor(x/rhs, y/rhs);
}
```