Tartalomjegyzék

Feladatok

Minden feladathoz nyiss új projektet IDE-ben vagy írd új file-ba ha parancssorból dolgozol!

Mostantól .cpp kiterjesztésû file-okban dolgozzunk!

Complex osztály operátorokkal

Módosítsuk az elõadás "http://wiki.math.bme.huComplex"http://wiki.math.bme.hu osztályát, hogy operator+, operator* is mûködjön, valamint tudjunk cout-al kiírni komplexeket.

Emlékeztetõül:

```
Complex operator+(Complex other);
friend ostream& operator<<(ostream& os, Complex& c);
```

```
#include<iostream>
#include<cmath>

using namespace std;


class Complex {
private:
  float re;
  float im;
public:
  Complex();
  Complex(const Complex& other);
  Complex(float r);
  Complex(float r, float i);

  Complex add(Complex other);
  Complex times(Complex other);
  float abs();

  void print();

  ~Complex();
};

Complex::Complex() {
  re = 0;
  im = 0;
}

Complex::Complex(const Complex& other) {
  re = other.re;
```

```
  im = other.im;
}

Complex::Complex(float r) {
  re = r;
  im = 0;
}

Complex::Complex(float r, float i) {
  re = r;
  im = i;
}

Complex Complex::add(Complex other) {
  return Complex(this->re + other.re, this->im + other.im);
}

Complex Complex::times(Complex other) {
  float real = this->re * other.re - this->im * other.im;
  float imag = this->re * other.im + this->im * other.re;
  return Complex(real, imag);
}

float Complex::abs() {
  return sqrt(this->re * this->re + this->im * this->im);
}

void Complex::print() {
  cout << re << " + " << im << "i" << endl;
}

Complex::~Complex() {
}

int main(void) {
  Complex a;
  Complex b = Complex(1,2);
  Complex c = a.times(b);

  a.print();
  b.print();
  c.print();

  (b.add(c)).print();

  cout << b.abs() << endl;

  return 0;
}
```

String konkatenálás

Írjuk meg a korábbi gyakon már említett, de közösen meg nem oldott C string konkatenáló függvényt. Most oldjuk ezt meg oszályon kívül, csak a függvényt.

Tehát a cél, hogy ha a függvényünk kap két string-et, akkor adjon vissza egy új stringet aminek az eleje az els? teljes string, a vége a második teljes string.

Pl:

```
char s1[] = "kis";
char s2[] = "kutya";
```

```
char* s3 = concat(s1, s2);
cout << s3 << endl;  // kiskutya-t kellene kiirnia
```

Próbáljuk lépésr?l lépésre megközelíteni a problémát!

- Megfelel? méret? char tömböt kell lefoglalnunk, ehhez kell tudni a 2 kapott string méretét. Írjunk meg egy string hossz-t számoló függvényt!
- Használva a 2 string hosszát hozzuk létre dinamikusan az új string-et. Dinamikusan kell (new-val), mert különben törl?dik, ha vége a függvénynek.
- Egy for ciklussal másoljuk az els? stringet az elejére.
- Majd egy másik for ciklussal másoljuk a másodikat ezután, vigyázzunk az indexelésre, itt ugye elcsúszik az új string és a második string indexelése.
- Ne felejtsük el a végére a lezáró nullát.

String oszály

Írjunk meg most String osztályt, amit lehet konstruálni C stringb?l és az operator+ metódusa a fenti konkatenálást végzi el két string-en.

Dictionary

Írjunk osztályt Dictionary néven, ami python szótárat imitál. A kulcsok string-ek legyenek. Az értékek egészek.

Ez hasznos lehet, egy példa dinamikusan létrehozott két dimenziós char tömbre:

```
char** keys = new char*[100];
for(int i = 0; i < 100; i++) {
  keys[i] = new char[20];
}
```

Az nem gond, ha a kulcsok hosszának van fels? határa, de próbáljuk úgy megoldani, hogy akárhány elemet behelyezhessünk a szótárba.

Példa a operator[]-ra:

```
int& operator[](int index);
```

A mi esetünkben a paraméter char* lenne (C string).

Fontos, hogy referenciát adunk vissza az operator[]-ban, így lehet balértékként is használni a metódus visszatérési értékét. Például:

```
dict["batman"] = 1;
```

Ezek legyenek meg:

- Default constructor: üres szótár
- operator[]: lehessen új elemeket hozzáadni ezzel (azaz m?ködjön jobb és bal értékként is). Ha meghívjuk egy kulccsal amihez nem tartozik még érték, akkor hozza létre ezt a kulcs/érték párt.
- Lehessen cout-olni. Amikor kiírjuk akkor látszódjon a teljes tartalma (minden pár). A formázás nem lényeges.
- remove(char*): törölje az adott kulcshoz tartozó kulcs/érték párt.
- Destructor

Megoldhatjátok dinamikusan növekv? tömbbel is, de sokkal szebb megoldás a korábban megoldott láncolt listát módosítani, hogy párokat tároljon és azt használni.