**Projekt - opis**

Uwaga: Projekt składa się z 3 części. Dokładna data oddania projektu jest w kalendarzu.  
  
  
Co przesyłamy:

* Pliki projektowe utworzone w Visual C++ lub Makefile (dla osób piszących pod LINUXEM) wraz z plikami źródłowymi cpp i h.
* Cały komplet plików powinien być spakowany zip-em i nazwany Imie\_Nazwisko\_indeks\_grupa\_punkty.zip. Punkty w nazwie pliku określają, na jaką ilość punktów pisany był projekt, czyli 30 lub 20.
* **Uwaga: jeżeli projekt nie uwzględnia liczb ujemnych to możemy dostać maksymalnie 20 punktów (8 za część pierwszą, 6, za drugą i 6 za trzecią)**
* **Uwaga: jeżeli projekt uwzględnia liczby ujemne to możemy dostać maksymalnie 30 punktów (18 za część pierwszą, 6 za drugą i 6 za trzecią)**
* Deklaracje każdej implementowanej klasy powinny być zapisane w osobnych plikach nagłówkowych z rozszerzeniem h, natomiast ich definicje w osobnym pliku z rozszerzeniem cpp.
* Odpowiednia funkcja main (zależnie od numeru części projektu) powinna znajdować się w pliku main1.cpp, main2.cpp, main3.cpp.
* Jeżeli czegoś w projekcie nie zrobiliśmy, coś nie działa, proszę umieścić plik bug.txt, w którym taką informację można umieścić.

Oceniane będą:  
poprawność działania.

* podział kodu na pliki projektowe (h, cpp).
* styl pisania i komentarze - im łatwiej się go czyta i im więcej się z niego rozumie tym ocena będzie wyższa, a im bardziej zagmatwany tym niższa.
* samodzielność – jeżeli dwa projekty będą zbyt podobne do siebie, oba zostaną nie zaliczone, bez dochodzenia, który z projektów był wersją źródłową i nie będzie mnie interesować, kto jest faktycznym autorem projektu (dotyczy to również Internetu na co proszę zwrócić uwagę).
* jeżeli będzie tego wymagała sytuacja, autor projektu może zostać poproszony o obronienie całego projektu, lub tylko poszczególnych jego części.
* **Projekt część 1**
* Napisz klasę umożliwiającą wykonywanie obliczeń na wielkich liczbach całkowitych.
* Implementacja za maksymalną ilość punktów powinna być zrobiona z wykorzystaniem poniższej klasy (uwaga nie można wykorzystywać żadnej konwersji na inne klasy i wszystkie operacje muszą być realizowane na tablicach):
* class DuzaLiczba  
       char \*wartosc;  
       .....  
     public:  
       .....
* Za mniejszą ilość punktów (-5) można wykorzystać poniższą klasę.
* class DuzaLiczba  
       string wartosc;  
       .....  
     public:  
       .....
* Liczby reprezentowane są jako łańcuchy znaków o dowolnej długości. Klasę należy zaimplementować tak, aby wykonanie poniższej funkcji main było poprawne. Należy pamiętać o konieczności implementacji wszystkich niezbędnych konstruktorów, operatorów, destruktorów, oraz o przedefiniowaniu wszystkich operatorów co jest istotą tej części projektu.
* Jeżeli w chwili obecnej nie wiesz co i jak zdefiniować, to większość wątpliwości zostanie "rozwiana" na tych i na następnych zajęciach (mam taką nadzieję).
* #include "DuzaLiczba.h"   
    
  int main(){  
  // konstruktor bezargumentowy inicjalizuje obiekt wartością 0  
  DuzaLiczba l1;  
  // konstruktor z argumentem typu int inicjalizuje obiekt wartością stało-liczbową  
  DuzaLiczba l2(9999);  
  // konstruktor z argumentem typu char\* inicjalizuje obiekt podaną wartością  
  DuzaLiczba l3("999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999");   
  // konstruktor kopiujący głęboko  
  DuzaLiczba l4(l3);   
  // też konstruktor kopiujący głęboko  
  DuzaLiczba l5 = l4;   
  DuzaLiczba l6(-987654);  
    
  DuzaLiczba \*l7 = new DuzaLiczba(“-999999999999999999999999999999”);  
    
  // Wszystkie obiekty od l1 do l6 są obiektami statycznymi,   
  // tzn. istnieją od początku do końca działania funkcji main   
  // natomiast obiekt wskazywany przez wskaźnik l7 jest   
  // obiektem dynamicznym, tzn. może być usunięty przez wywołanie   
  // operatora delete.  
    
    
  // przedefiniowany operator >>   
  cin >> l1;   
  // przedefiniowany operator <<  
  cout << l1 << l2 << l3 << l4 << l5 << l6 << \*l7;   
    
  // usunięcie obiektu wskazywanego przez l7  
  delete l7;   
    
  // przedefiniowany operator =  
  l1 = l2;   
  // przedefiniowane operatory +-\*/ dzielenie całkowite  
  l1 = l1 + l2 - l3 \* l4 / l5;   
    
  // przedefiniowany operator +=  
  l1 += 23;   
  // przedefiniowany operator +=   
  l1 += l4;  
  // przedefiniowany operator -=  
  l2 -= l3;  
  // przedefiniowany operator \*=  
  l3 \*= l4;  
    
  // przedefiniowany operator ==  
  if(l4 == "23424")  
  // przedefiniowany operator /= dzielenie całkowite  
  l4 /= l5;   
    
  // przedefiniowany operator !=  
  if(l5 != l4) {  
  // przedefiniowany jednoargumentowy operator -  
  l5 = -l5;  
  cout << l5;  
  }  
    
  // przedefiniowany operator =  
  l2 = "1221";  
    
  DuzaLiczba silnia = 0;  
  // przedefiniowane operatory >= <  
  if(l2 >= 0 && l2 < "1223")  
  {   
  // przedefiniowany operator !  
  silnia = !l2;   
  cout << silnia ;  
  }  
    
  return 0;  
  }
* **Projekt część 2**
* Napisz klasę umożliwiającą wykonywanie obliczeń na wielkich liczbach zespolonych.
* class DuzaLiczbaZespolona
* DuzaLiczba \*real;
* DuzaLiczba \*imagine;
* .....
* public:
* .....
* Liczby zespolone reprezentowane jako obiekty klasy DuzaLiczbaZespolona, której dwoma podstawowymi polami są pola real i imagine, których znaczenie od strony matematycznej niestety trzeba poznać samodzielnie, natomiast od strony programistycznej  są to po prostu wskaźniki na obiekty klasy DuzaLiczba.
* Klasę należy zaimplementować tak, aby wykonanie poniższej funkcji main było poprawne. Należy pamiętać o konieczności implementacji wszystkich niezbędne konstruktorów, operatorów, destruktorów, oraz o przedefinowaniu wszystkich operatorów co jest istotą tej części projektu.
* Uwaga: jeżeli w poniższej funkcji należy przedefiniować jakiś operator, który wymaga jednej liczby zespolonej i jednej całkowitej, jak np.: l4=="23424" wówczas w operacji porównania bierze udział liczba 23424 i część rzeczywista liczby l4.
* #include "DuzaLiczba.h"
* #include "DuzaLiczbaZespolona.h"

* int main(){
* // konstruktor bezargumentowy inicjalizuje obiekt wartością (0,0)
* DuzaLiczbaZespolona l1;
* // konstruktor z argumentem typu int inicjalizuje obiekt
* // wartością stało-liczbową (172834,0)
* DuzaLiczbaZespolona l2(172834);
* // konstruktor z dwoma argumentami typu int inicjalizuje
* // inicjalizuje obiekt wartością (134,394)
* DuzaLiczbaZespolona l2a(134, 394);
* // konstruktor z argumentem typu char\* inicjalizuje obiekt podaną
* // wartością (“1728390”)
* DuzaLiczbaZespolona l3("1728390");
* // konstruktor z argumentem typu char\* inicjalizuje obiekt
* // podaną wartością (17283,-420)
* DuzaLiczbaZespolona l3a("17283", "-420");
* // konstruktor kopiujący głęboko
* DuzaLiczbaZespolona l4(l3);
* // konstruktor z argumentem typu char\* inicjalizuje
* DuzaLiczbaZespolona l5=l4;
* DuzaLiczbaZespolona l6(-172834);
* DuzaLiczbaZespolona \*l7 = new DuzaLiczbaZespolona("-172839023498234792834798237494");
* // Wszystkie obiekty od l1 do l6 są obiektami statycznymi,
* // tzn. istnieją od początku do końca działania funkcji main
* // natomiast obiekt wskazywany przez wskaźnik l7 jest obiektem
* // dynamicznym, tzn. może być usunięty przez wywołanie operatora delete.
* // przedefiniowany operator >>
* cin >> l1;
* // przedefiniowany operator <<
* cout << l1 << l2 << l3 << l4 << l5 << l6 << \*l7;
* cout << l2a << l3a;
* // usunięcie obiektu wskazywanego przez l7
* delete l7;
* // przedefiniowany operator =
* l1 = l2;
* // przedefiniowane operatory +-\*/  dzielenie całkowite
* l1 = l1 + l2 - l3 \* l4 / l5;
* // przedefiniowany operator += - zwiększenie składowej rzeczywistej liczby l1 o 23
* l1 += 23;
* // przedefiniowany operator +=
* l1 += l4;
* // przedefiniowany operator -=
* l2 -= l3;
* // przedefiniowany operator \*=
* l3 \*= l4;
* // przedefiniowany operator ==
* if(l4 == "23424")
* // przedefiniowany operator /=  dzielenie całkowite
* l4 /= l5;
* // przedefiniowany operator ==
* if(l4 == l5)
* // przedefiniowany operator /=  dzielenie całkowite
* l4 /= l5;
* // przedefiniowany operator !=
* if(l5 != l4) {
* // przedefiniowany operator – zmiana wartości składowych na przeciwne
* l5 = -l5;
* cout << l5;
* }
* // przedefiniowany operator =
* l2 = "1221";
* // przedefiniowane operatory >=  <
* if(l2 >= 0 && l2 < "1223"){
* cout << "ok" ;
* }
* return 0;
* }

### Projekt część 3

### 

### Lista wielkich liczb i wielkich liczb zespolonych.

Napisz klasę umożliwiającą utworzenie dwukierunkowej listy, która jednocześnie będzie w stanie przechowywać zarówno obiekty reprezentujące duże liczby, jak i obiekty reprezentujące liczby zespolone. Jak to zrobić? Jest takie pojęcie "dziedziczenie", które należy wykorzystać do tego celu, ale o tym więcej już na ćwiczeniach, albo konsultacjach.

Na liście możemy wykonać następujące operacje:

* wstaw element na koniec listy
* wstaw element na początek listy
* wstaw element na i-tą pozycję, jeżeli podano pozycję ujemną to wstaw element na początek, a jak podano pozycję, która nie istnieje to wstaw na koniec.
* usuń ostatni element listy
* usuń pierwszy element listy
* usuń i-ty element listy
* usuń całą listę
* posortuj listę rosnąco
* wyświetl listę (operator<<).

Uwaga, sortowanie elementów listy odbywa się w następujący sposób: porównywane są wszystkie liczby bez względu na klasę,  istotne jest aby w klasach DuzaLiczba iDuzaLiczbaZespolona była zdefiniowana funkcja o nazwie Wartosc zwracająca wartość typu char\*, która będzie porównana, czyli:

DuzaLiczba \*l1 = new DuzaLiczba("1500"));

DuzaLiczbaZespolona  \*l2 = new DuzaLiczbaZespolona(-3000,1499);

cout << l1->Wartosc();  // "1500"

cout << l2->Wartosc();  // "3000"

lub np.:

cout << l2->Wartosc();  // "1499"

lub np.:

cout << l2->Wartosc();  // "-1501"

I później….

if(Lista::porownajLiczby(l1->Wartosc(),l2->Wartosc()))

// zamień miejscami

else

// nie zamieniaj miejscami

lub nawet tak, a wywołanie funkcji  Wartosc  zostanie wykonane wewnątrz funkcji, a nie na zewnątrz jak poprzednio:

if(Lista::porownajLiczby(l1,l2))

// zamień miejscami

else

// nie zamieniaj miejscami

Uwaga, funkcja porownajLiczby jest funkcją statyczną. Co to znaczy?

Aby poprawnie rozwiązać zadanie należy zaimplementować dwie klasy. Pierwsza z nich:

 class ListaDuzychLiczb {

  ElementListy \*pierwszy;

  ElementListy \*ostatni;

 public:

   .....

 };

reprezentuje całą listę wskazując jej pierwszy i ostatni element, wraz ze zdefiniowanymi powyższymi operacjami. Druga klasa reprezentuje pojedynczy element listy (ElementListy) wraz ze wskaźnikami na element poprzedni i element następny na liście

class ElementListy {

 Liczba \*liczba;

 ElementListy \*nastepnyElement;

 ElementListy \*poprzedniElement;

public:

  .....

};

Co to za klasa Liczba i skąd ona się tu wzięła? Mianowicie, klasa ta jest klasą bazową (dziedziczenie) dla klasDuzaLiczba i DuzaLiczbaZespolona. Ale więcej o tym na ćwiczeniach.

Klasy należy zaimplementować tak, aby wykonanie poniższej funkcji main było poprawne. Należy pamiętać o konieczności implementacji wszystkich niezbędnych konstruktorów, operatorów, destruktorów.

int main(){

 ListaDuzychLiczb \*lista = new ListaDuzychLiczb();

 lista->wstawNaPoczatek(new DuzaLiczba("9129081391"));

 lista->wstawNaPoczatek(new DuzaLiczbaZespolona(12908,1391));

 lista->wstawNaKoniec(new DuzaLiczba("81391"));

 lista->wstawNaKoniec(new DuzaLiczbaZespolona(12908,1391));

 for(int i=0; i<15; i++){

   lista->wstawNaPozycje (new DuzaLiczba(rand()\*9283), 1);

   lista->wstawNaPozycje (new DuzaLiczbaZespolona(rand()\*9293, rand()\*92983), 1);

 }

 cout << lista;

 lista->sortuj();

 cout << lista;

 lista->usunPierwszyElement();

 cout << lista;

 lista->usunOstatniElement();

 cout << lista;

 lista->usunPozycje(3);

 cout << lista;

 lista->usunWszystkieElementy();

 cout << lista;

 delete lista;