Zadanie Wielomiany

Autor: Daniel Wilczak

Wprowadzenie.

Państwa zadaniem jest implementacja dwóch klas realizujących pewne operacje algebraiczne. Są to:

- **Z3** - klasa dostarczająca implementacji operacji w ciele liczbowym Z_3
- **Polynomial** - klasa dostarczająca implementacji na wielomianach dowolnego stopnia o współczynnikach Z_3

Nazwy plików.

Całość implementacji musi być podzielona na 4 pliki (ważna jest wielkość liter):

- **Z3.h** i **Z3.cpp**
- **Polynomial.h** i **Polynomial.cpp**

Pliki nagłówkowe ***.h** powinny dostarczać definicji wszystkich wymaganych klas oraz deklaracji funkcji globalnych (szczegóły niżej).
Pliki ***.cpp** powinny dostarczać implementacji tych funkcjonalności.
Każdy plik ***.cpp** będzie niezależnie kompilowany, a całość będzie linkowana z programem testującym zawierającym funkcję **main**. Dlatego nadesłane przez Państwa rozwiązania **nie mogą zawierać funkcji main**.

Uwaga: Każdy z wymienionych plików musi mieć w pierwszej linii w komentarzu imię i nazwisko autora rozwiązania, np.

```
// Daniel Wilczak
```

Sposób przesłania rozwiązania.

Do systemu BaCa należy przesłać jeden plik: archiwum zip zawierające dokładnie 4 pliki o nazwach jak wyżej, bez podkatalogów.

Szczegółowe wymagania dotyczące implementacji klas.

- **w rozwiązaniach nie wolno korzystać z kolekcji STL**, np. z <vector>.
  Rozwiązania wykorzystujące pliki nagłówkowe z biblioteki standardowej inne niż
  - **<iostream>**,
  - **<sstream>**,
  - **<string>**,
  - **<cstdlib>** oraz

•<cstring>

będą automatycznie odrzucone.

•Wszystkie klasy powinny poprawnie zarządzać pamięcią, w szczególności konstruktory kopiujące, destruktory i operatory przypisania powinny być zdefiniowane wszędzie tam, gdzie jest to konieczne.

•Wszędzie gdzie ma to sens należy akceptować stałe obiekty oraz tablice obiektów stałych jako argumenty funkcji. W szczegółowych opisach poniżej nie będzie to sprecyzowane.

Wymagany publiczny interfejs klasy **Z3**.

Klasa Z2 powinna dostarczać następujących publicznych metod:

Konstruktor domyślny
: tworzy liczbę zero

Konstruktor z argumentem typu short int
: tworzy liczbę Z_3 na podstawie podanego argumentu

Operator konwersji do short int
: zwraca liczbę naturalną ze zbioru {0,1,2}

operator+=
: realizuje operację **a+=b** dla liczb w ciele Z_3. Zwraca referencję do uaktualnionego obiektu **a**.

operator-=
: realizuje operację **a-=b** dla liczb w ciele Z_3. Zwraca referencję do uaktualnionego obiektu **a**.

operator*=
: realizuje operację **a*=b** dla liczb w ciele Z_3. Zwraca referencję do uaktualnionego obiektu **a**.

operator/=
: realizuje operację **a/=b** dla liczb w ciele Z. Zwraca referencję do uaktualnionego obiektu **a**. W przypadku dzielenia przez zero należy wypisać na standardowe wyjście łańcuch

```
"Dzielenie przez zero\n"
```

i zwrócić niezmieniony obiekt **a**.

Ponadto należy zdefiniować następujące operatory jako **funkcje globalne**:

operator+
: zwraca obliczony wynik **a+b** jako obiekt **Z3**.

operator-
: zwraca obliczony wynik **a-b** jako obiekt **Z3**.

operator*
: zwraca obliczony wynik **a*b** jako obiekt **Z3**.

operator/
: zwraca obliczony wynik **a/b** jako obiekt **Z3**. W przypadku dzielenia przez zero należy wypisać na standardowe wyjście łańcuch

```
"Dzielenie przez zero\n"
```

i zwrócić niezmieniony obiekt **a**.

operator<<
realizuje wypisanie liczby **Z3** do strumienia **std::ostream** jako liczby całkowitej ze zbioru {0,1,2}

Wymagany publiczny interfejs klasy **Wielomian**.

Klasa `Wielomian` powinna dostarczać następujących publicznych metod:

Konstruktor domyślny
tworzy wielomian stopnia zero równy zero

Konstruktor z argumentem typu unsigned int oraz tablicą obiektów Z3
tworzy Wielomian o długości podanej jako pierwszy argument o współczynnikach z podanej tablicy. Należy przyjąć założenie, że tablica zawiera wystarczającą liczbę elementów. Zakładamy również, że współczynniki są podane w kolejności od wyrazu wolnego, do współczynnika przy najwyższej potędze.

operator[]
zwraca współczynnik wielomianu przy **i**-tej potędze. Indeks jest liczbą typu **unsigned int**. W przypadku, gdy podany indeks jest niepoprawny (większy od stopia wielomianu), należy wypisać na standardowe wyjście łańcuch

```
"Niepoprawny indeks wielomianu\n"
```

oraz zwrócić współczynnik przy zerowej potędze wielomianu.

operator+=
realizuje operację **a+=b** dla wielomianów. Zwraca referencję do uaktualnionego obiektu **a**.

operator-=
realizuje operację **a-=b** dla wielomianów. Zwraca referencję do uaktualnionego obiektu **a**.

operator*=
realizuje operację **a*=b** dla wielomianów. Zwraca referencję do uaktualnionego obiektu **a**.

operator*=
realizuje operację **a*=s**, gdzie **a** jest wielomianem, a **s** jest liczbą Z_3. Zwraca referencję do uaktualnionego obiektu **a**.

operator/=
realizuje operację **a/=s**, gdzie **a** jest wielomianem, a **s** jest liczbą Z_3. Zwraca referencję do uaktualnionego obiektu **a**. W przypadku dzielenia przez zero należy wypisać na standardowe wyjście łańcuch

```
"Dzielenie przez zero\n"
```

i zwrócić niezmieniony obiekt **a**.

degree
bezargumentowa funkcja zwracająca stopien wielomianu jako liczbę typu **unsigned int**.

toString

funkcja o argumencie **xVar** typu **std::string** i wartości typu **std::string**. Zwraca wzór wielomianu w postaci sumy kolejnych potęg zmiennej **xVar** w kolejności od wyrazu wolnego do najwyższej potęgi. Ponadto należy:

- wielomian zerowy wypisać jako **0**. Jest to wyjątek, poza którym nie wypisujemy współczynników zerowych.
- jeśli wielomian nie jest zerowy, to wyraz wolny wypisujemy tylko wtedy, gdy jest niezerowy i jako liczbę **1** lub **2**
- pozostałe jednomiany wypisujemy jako **s*xVar^i** o ile współczynnik **s** przy danej potędze jest niezerowy

Powyższe łączymy znakiem **+**. Przykłady poprawnych wartości dla wywołania **toString("x")**:

- **0**
- **1**
- **2*x^1**
- **1+2*x^7+1*x^1234**

Ponadto należy zdefiniować następujące operatory jako **funkcje globalne**:

operator+, operator-, operator*

zwracają sumę, różnicę i iloczyn wielomianów **a+b**, **a-b** oraz **a*b**.

operator*, operator/

mnożenie i dzielenie przez liczbę **s*a** oraz **a/s**, gdzie **s** jest liczbą. W przypadku dzielenia przez zero należy wypisać na standardowe wyjście łańcuch

```
"Dzielenie przez zero\n"
```

i zwrócić niezmieniony obiekt **a**.

mod

czteroargumentowa funkcja globalna o argumentach kolejno **u,v,q,r**. Jej zadaniem jest obliczenie wyniku z dzielenia i reszty z dzielenia wielomianów **u** i **v**. Wyniki obliczeń są zapisane w argumentach **q (wynik), r (reszta)** przesłanych do funkcji przez referencję. Obliczone wielomiany **q,r** mają spełniać tożsamość
u = q*v + r
przy czym stopień **r** jest mniejszy niż stopień **v**. Wyjątkiem jest dzielenie przez wielomian stopnia zero **v=1** lub **v=2**, gdzie reszta też jest stopnia zero i jest równa **r=0**.
W przypadku dzielenia przez wielomian zerowy należy wysłać do standardowego strumienia łańcuch

```
"Dzielenie przez zero\n"
```

i pozostawić argumenty **q,r** niezmienione.

operator<<

realizuje wypisanie wielomianu **a** do strumienia w następującym formacie

$$\{a_0,a_1,...,a_n\}$$

gdzie **n** jest stopniem wielomianu, a_i to współczynnik przy **i**-tej potędze.

operator>>

realizuje wczytanie wielomianu **a** z podanego strumienia. Operator ma ignorować wszystkie znaki strumienia aż do pierwszego napotkanego nawiasu **'{'**. Wtedy rozpoczyna czytanie kolejnych współczynników wielomianu aż do napotkania

zamykającego nawiasu **'}'**. Zakładamy, że w strumieniu wejściowym jest wielomian w formacie zgodnym z tym, co zostało wydrukowane za pomocą **operator<<** (z dokładnością do pewnej liczby początkowych nieokreślonych znaków poprzedzających **'{'**).