Indywidualny projekt programistyczny (Info, I rok) 16/17

NAWIGACJA

Kokpit
- Strona główna

Strony

Moje kursy
- MD.INFO.I.17/18
- IO.INFO.II.17/18
- BD.INFO.II.17/18
- JNP1.INFO.II.17/18
- WPI.LAB.INFO.I.16/17
- WPI.INFO.I.16/17
- IPP.INFO.I.16/17
  - Uczestnicy
  - Odznaki
  - Kompetencje
  - Oceny
  - Główne składowe
  - Temat 1
  - Temat 2
  - Temat 3
  - Temat 4
  - Temat 5
  - Temat 6
  - Temat 7
  - Temat 8

Wielomiany, część 1

Zadanie wielomiany

Tegoroczne duże zadanie polega na zaimplementowaniu operacji na wielomianach rzadkich wielu zmiennych.

Wielomian jest sumą jednomianów px^n, gdzie x jest nazwą zmiennej, n jest wykładnikiem będącym nieujemną liczbą całkowitą, a p jest współczynnikiem. Wykładniki jednomianów wchodzących w skład wielomianu są parami różne. Wielomian, który nie jest stały, nazywamy normalnym. Współczynnik jednomianu jest wielomianem nowej zmiennej, innej niż x, i innej niż wszystkie dotychczas użyte w tym jednomianie zmienne. Innymi słowy, jeśli oznaczymy zmienne wielomianu przez x_0, x_1, x_2 itd., to wtedy współczynnik jednomianu o zmiennej x_i, jeśli jest wielomianem normalnym, to jest on wielomianem zmiennej x_{i+1}. Wielomiany są rzadkie, co oznacza, że stopień wielomianu może być znacznie większy niż liczba składowych jednomianów.

Zadanie wielomiany część 1

Jako pierwszą część zadania należy zaimplementować bibliotekę podstawowych operacji na wielomianach rzadkich wielu zmiennych. Opis funkcji znajduje się w pliku `poly.h` w formacie komentarzy dla programu doxygen.

Dostarczamy

W repozytorium `https://git.mimuw.edu.pl/IPP-login.git` (gdzie login to identyfikator używany do logowania się w laboratorium komputerowym) znajduje się wstępna implementacja rozwiązania tego zadania. Znajdują się tam następujące pliki:

- `src/poly.h` – deklaracja interfejsu biblioteki wraz z jej dokumentacją w formacie doxygen
- `src/test_poly.c` – testy biblioteki
- `CMakeLists.txt` – plik konfiguracyjny programu cmake
- `Doxyfile.in` – plik konfiguracyjny programu doxygen
- `MainPage.dox` – strona główna dokumentacji w formacie doxygen

Zastrzegamy sobie możliwość wgrania poprawek do tego szablonu. Będziemy

Wielomiany, część 1
CMake
Doxygen
Temat 9
Temat 10
Temat 11
Temat 12
Temat 13
Temat 14
MD.INFO.I.16/17
PO.INFO.I.16/17

ADMINISTRACJA
Administracja kursem

je wgrywać do gałęzi `template/part1`. Lista poprawek:

- *24 kwietnia, 15:17* - dodany brakujący `const` i poprawienie komentarzy,
- *6 maja, 22:47* - pierwsza wersja testów,
- *7 maja, 14:42* - poprawki w testach, błędny `PolyDestroy` i używanie `MonoFromPoly` w `SimpleAddMonosTest`,
- *8 maja, 11:14* - poprawki bugów i dodana funkcja `MemoryTest`,
- *9 maja, 0:16* - poprawki w testach, brakujące `PolyDestroy` i większy test `TestMul2`,
- *13 maja, 14:05* - połączenie testów `deg` i `deg-by`.

Wymagamy

Jako rozwiązanie części 1 zadania wymagamy:

- uzupełnienia implementacji w pliku `src/poly.h`
- stworzenia pliku `src/poly.c` z implementacją wymaganych funkcji
- uzupełnienia dokumentacji

Gotowe rozwiązanie powinno się kompilować za pomocą sekwencji poleceń:

```
mkdir release
cd release
cmake ..
make
make doc
```

W wyniku wykonania tych poleceń powinien powstać plik wykonywalny `test_poly` oraz dokumentacja. W poleceniu `cmake` powno być również możliwe jawne określenie wariantu budowania pliku wynikowego:

```
cmake -D CMAKE_BUILD_TYPE=Release ..
cmake -D CMAKE_BUILD_TYPE=Debug ..
```

Zawartość pliku `src/poly.h` i innych dostarczonych przez nas plików można modyfikować, o ile nie zmienia to interfejsu biblioteki i zachowuje wymagania podane w treści zadania, przy czym nie należy zmieniać opcji kompilacji.

Wymagania dotyczące implementacji

Konwencje interfejsu

W interfejsie zostały przyjęte pewne konwencje, które mają ułatwić zarządzanie pamięcią. Dzięki tym konwencjom wiadomo, kto jest właścicielem obiektu. Bycie właścicielem obiektu implikuje odpowiedzialność za zwolnienie go z pamięci. W przypadku struktur `Poly` i `Mono` zwalnianie pamięci uzyskuje się poprzez wywołania funkcji odpowiednio `PolyDestroy` i `MonoDestroy`.

Podstawową konwencją jest *przekazywanie argumentów przez zmienną*. W języku C do tego celu użyty jest typ wskaźnikowy (np. `const Poly *`). Kod wołający funkcję, której przekazujemy argument przez zmienną, jest odpowiedzialny za utworzenie odpowiedniego wskaźnika. Może to być wskaźnik na lokalną zmienną, bądź też wskaźnik uzyskany w wyniku alokacji pamięci (np. przez `malloc`). W tym drugim wypadku trzeba pamiętać, aby

zwolnić tę pamięć. Odpowiedzialność za zwolnienie tego wskaźnika nigdy nie przechodzi na wołaną funkcję.

Przy niektórych funkcjach *argumenty przechodzą na własność* funkcji wołanej. Jest to zaznaczone w komentarzu opisującym daną funkcję. Funkcja przejmuje zawartość pamięci wskazywanej przez przekazany wskaźnik. Zazwyczaj jest to pojedyncza struktura (np. `Poly`, `Mono`) bądź tablica struktur.

Wynikiem niektórych funkcji jest struktura (np. `Poly`, `Mono`). Przyjmujemy tu konwencję otrzymywania *wyniku na własność*, co oznacza, że kod wołający taką funkcję otrzymuję zwracaną wartość na własność.

Przykłady przekazywania własności i zwalniania pamięci przez ostatniego właściciela:

```
{
    Poly p1 = ...
    PolyDestroy(&p1);
}
{
    Poly p1 = ...
    Mono m1 = MonoFromPoly(&p1, 7); // przekazanie własności p1
    MonoDestroy(&m1);
}
{
    Poly p1 = ...
    Mono m1 = MonoFromPoly(&p1, 7); // przekazanie własności p1
    Poly p2 = PolyAddMonos(1, &m1); // przekazanie własności m1
    PolyDestroy(&p2);
}
```

Złożoność

Złożoność operacji na wielomianach, poza mnożeniem, powinna być liniowa względem wielkości wielomianu, a mnożenia liniowa względem iloczynu wielkości mnożonych wielomianów. Wielkość wielomianu jest liczbą jednomianów liczoną po wszystkich zmiennych. Tam, gdzie to jest możliwe, operacje powinny działać w czasie stałym.

Wytyczną jest przejście testów i zmieszczenie się w limitach czasowych. Najprawdopodobniej zostaną użyte następujący limity czasowe:

- 30s dla `long-polynomial`,
- 3s dla `add-req` i `sub-req`,
- 2s dla `mul` oraz
- 1s dla pozostałych testów.

Inne

Niepowodzenie alokacji pamięci należy wykrywać za pomocą asercji.

Oddawanie rozwiązania

Rozwiązanie należy oddawać przez wspomniane wyżej repozytorium git. W repozytorium mają się znaleźć wszystkie pliki niezbędne do zbudowania pliku

wykonywalnego `test_poly` oraz dokumentacji. *W repozytorium nie wolno umieszczać plików binarnych ani tymczasowych.* W Moodle jako rozwiązanie należy umieścić tekst zawierający identyfikator commita z finalną wersją rozwiązania, na przykład:

```
Finalna wersja mojego rozwiązania części 1 zadania wielomiany zna
jduje
się w repozytorium w commicie 518507a7e9ea50e099b33cb6ca3d3141bc1
d6638.
```

Rozwiązanie należy zatwierdzić (git commit) i wysłać do repozytorium (git push) najpóźniej do godz. 23.55 dnia 13 maja 2017.

Punktacja

Za w pełni poprawne rozwiązanie zadania implementujące wszystkie wymagane funkcjonalności można zdobyć maksymalnie 20 punktów. Od tej oceny będą odejmowane punkty karne za poniższe uchybienia:

- Za każdy test, którego biblioteka nie przejdzie, traci się 1 punkt.
- Za wycieki pamięci można stracić do 6 punktów.
- Za niezgodne ze specyfikacją nazwy plików w rozwiązaniu lub umieszczenie w repozytorium niepotrzebnych albo tymczasowych plików można stracić do 2 punktów.
- Za błędy w stylu kodowania można stracić do 3 punktów.
- Za braki w dokumentacji można stracić do 2 punktów.

Rozwiązania należy implementować *samodzielnie* pod rygorem niezaliczenia przedmiotu.

Status przesłanego zadania

Numer próby	To jest próba nr 1.
Status przesłanego zadania	Przesłane do oceny
Stan oceniania	Nie ocenione
Ostatnio modyfikowane	czwartek, 24 sierpień 2017, 13:03
Tekst online	(40 słów) EDIT POWAKACYJNY, 2017-08-24 Finalna wersja mojego rozwiązania części 1 zadania wielomiany znajduje się w repozytorium w commit ...
Komentarz do przesłanego zadania	Komentarze (0)

Jesteś zalogowany(a) jako Piotr Szuberski (Wyloguj)
IPP.INFO.I.16/17

Moodle, wersja 3.2.2+ | moodle@mimuw.edu.pl