

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

Kokpit ► Moje kursy ► IPP.INFO.I.16/17 ► Temat 9 ► Wielomiany, część 2

Wielomiany, część 2

Zadanie wielomiany część 2

Jako drugą część zadania należy zaimplementować kalkulator działający na wielomianach i stosujący odwrotną notację polską oraz skrypt bashowy automatyzujący wykonywanie zapisanych w plikach sekwencji operacji na wielomianach.

Kalkulator

Program kalkulatora czyta dane wierszami ze standardowego wejścia. Wiersz zawiera wielomian lub polecenie do wykonania.

Specyfikacja danych wejściowych

Wielomian reprezentowany jest na wejściu jako stała, jednomian lub suma jednomianów. Stała jest liczbą całkowitą. Jednomian reprezentowany jest jako $(coeff, exp)$, gdzie współczynnik $coeff$ jest wielomianem, a wykładnik exp jest liczbą nieujemną. Do wyrażenia sumy używany jest znak $+$. Jeśli wiersz zawiera wielomian, to program wstawia go na stos. Przykłady poprawnych wielomianów:

```
0
1
-2
(0,0)
(1,0)
(-2,0)
(1,1)
(1,0)+(1,2)
(1,2)+(1,0)
(1,2)+(-1,2)
(1,2)+(-2,2)
((1,2),15)+(-7,8)
(3,1)+((4,4),100),2)
```

Kalkulator wykonuje następujące polecenia:

- `ZERO` – wstawia na wierzchołek stosu wielomian tożsamościowo równy zeru;
- `IS_COEFF` – sprawdza, czy wielomian na wierzchołku stosu jest współczynnikiem – wypisuje na standardowe wyjście 0 lub 1;

- `IS_ZERO` – sprawdza, czy wielomian na wierzchołku stosu jest tożsamościowo równy zeru – wypisuje na standardowe wyjście 0 lub 1;
- `CLONE` – wstawia na stos kopię wielomianu z wierzchołka;
- `ADD` – dodaje dwa wielomiany z wierzchu stosu, usuwa je i wstawia na wierzchołek stosu ich sumę;
- `MUL` – mnoży dwa wielomiany z wierzchu stosu, usuwa je i wstawia na wierzchołek stosu ich iloczyn;
- `NEG` – neguje wielomian na wierzchołku stosu;
- `SUB` – odejmuje od wielomianu z wierzchołka wielomian pod wierzchołkiem, usuwa je i wstawia na wierzchołek stosu różnicę;
- `IS_EQ` – sprawdza, czy dwa wielomiany na wierzchu stosu są równe – wypisuje na standardowe wyjście 0 lub 1;
- `DEG` – wypisuje na standardowe wyjście stopień wielomianu (-1 dla wielomianu tożsamościowo równego zeru);
- `DEG_BY idx` – wypisuje na standardowe wyjście stopień wielomianu ze względu na zmienną o numerze `idx` (-1 dla wielomianu tożsamościowo równego zeru);
- `AT x` – wylicza wartość wielomianu w punkcie `x`, usuwa wielomian z wierzchołka i wstawia na stos wynik operacji;
- `PRINT` – wypisuje na standardowe wyjście wielomian z wierzchołka stosu w formacie akceptowanym przez parser;
- `POP` – usuwa wielomian z wierzchołka stosu.

Wypisywany poleceniem `PRINT` wielomian powinien mieć jak najprostszą postać. Wykładniki wypisywanych jednomianów nie powinny się powtarzać. Jednomiany powinny być posortowane rosnąco według wykładników. Podane wyżej wielomiany powinny zostać wypisane następująco:

```
0
1
-2
0
1
-2
(1,1)
(1,0)+(1,2)
(1,0)+(1,2)
0
(-1,2)
(-7,8)+((1,2),15)
(3,1)+((4,4),100),2)
```

Sprawdzanie poprawności danych wejściowych i obsługa błędów

Program nie powinien zakładać maksymalnej długości wiersza. Poprawny wiersz nie zawiera żadnych dodatkowych białych znaków oprócz pojedynczej spacji separującej parametr poleceń `AT` i `DEG_BY` od polecenia. Program wypisuje komunikaty o błędach na standardowe wyjście błędów. Poniżej `r` oznacza numer wiersza, `c` – numer kolumny, a `\n` – znak przejścia do nowego wiersza. Wiersze i kolumny numerujemy od 1.

Na potrzeby obsługi błędów uznajemy, że wiersz zawiera polecenie, jeśli zaczyna się od wielkiej lub małej litery alfabetu angielskiego, mimo że wszystkie polecenia zaczynają się od wielkiej litery. Jeśli wiersz nie zaczyna się od litery, należy spróbować w tym wierszu sparsować wielomian.

Jeśli na stosie jest za mało wielomianów, aby wykonać polecenie, program wypisuje:

```
ERROR r STACK UNDERFLOW\n
```

Jeśli program wykryje błąd podczas parsowania wielomianu, wypisuje numer kolumny zawierającej znak, który jako pierwszy powoduje, że nie jest to poprawny wielomian, niezależnie od tego, jakie są dalsze znaki. Jeśli wczytywana jest liczba, to pierwsza cyfra, która powoduje, że liczba jest niepoprawna (np. przekracza dopuszczalny zakres) jest błędnym znakiem. Jeśli wiersz się nagle kończy, bo wczytano znak przejścia do nowej linii, to będzie to kolumna, gdzie pojawił się ten znak nowej linii. Komunikat o błędzie parsowania wielomianu ma postać:

```
ERROR r c\n
```

Kolejne komunikaty o błędach dotyczą sytuacji, gdy parsowane jest polecenie.

Jeśli program wykryje niepoprawną nazwę polecenia, wypisuje:

```
ERROR r WRONG COMMAND\n
```

Jeśli w poleceniu `AT` nie podano parametru lub jest on niepoprawny, program wypisuje:

```
ERROR r WRONG VALUE\n
```

Jeśli w poleceniu `DEG_BY` nie podano parametru lub jest on niepoprawny, program wypisuje:

```
ERROR r WRONG VARIABLE\n
```

Wartość współczynnika lub parametru polecenia `AT` uznajemy za niepoprawną, jeśli jest ona mniejsza od `LONG_MIN` lub większa od `LONG_MAX`. Wartość wykładnika uznajemy za niepoprawną, jeśli jest ona mniejsza od 0 lub większa od `INT_MAX`. Wartość parametru polecenia `DEG_BY` uznajemy za niepoprawną, jeśli jest ona mniejsza od 0 lub większa od `UINT_MAX`.

Program może, ale nie musi, akceptować liczby z dodatkowymi wiodącymi zerami. Znak `+` służy tylko do wyrażania sumy jednomianów i nie może poprzedzać liczby.

Skrypt

Skrypt przyjmuje dwa parametry: nazwę programu i nazwę katalogu z danymi. Skrypt wypisuje informację o błędzie i zwraca 1, jeśli podane parametry są nieprawidłowe:

- zła liczba parametrów,
- podana nazwa programu nie wskazuje na plik wykonywalny,
- podana nazwa katalogu nie wskazuje na katalog.

Skrypt przetwarza iteracyjnie pliki znalezione w podanym katalogu w następujący sposób. Pierwszym plikiem jest plik zaczynający się linią z napisem `START`. Na końcu pliku jest linia z napisem `FILE name` lub `STOP`. Napis `FILE name` oznacza, że w kolejnej iteracji należy przetworzyć plik o nazwie `name`. Napis `STOP` oznacza, że należy zakończyć przetwarzanie. Na wejście programu kalkulatora podawane jest połączone wyjście programu z poprzedniej iteracji i zawartość następnego pliku. Dla pierwszego pliku poprzednie wyjście jest puste. W danych podawanych na wejście programu pomijane są linie z napisami `START`, `FILE` i `STOP`. Pozostałe linie są poleceniami dla programu kalkulatora. Po przetworzeniu ostatniego pliku skrypt wypisuje na standardowe wyjście wynik ostatniego uruchomienia kalkulatora i zwraca 0. Zakładamy, że w podanym katalogu

- istnieje przynajmniej jeden plik,
- pliki mają format zgodny ze specyfikacją,

- w plikach dane dla kalkulatora są poprawne – skrypt nie musi obsługiwać błędów wypisywanych przez kalkulator.

Dostarczamy

Rozwiązanie części 2 zadania powinno korzystać z własnego rozwiązania części 1. Istnieje możliwość zaimplementowania części 2 zadania, bazując na otrzymanej od nas implementacji części 1. Trzeba jednak o to poprosić (mail do Jakuba Pawlewicza) i w takim przypadku ocena z części 1 zostanie zmniejszona o połowę.

W późniejszym terminie repozytorium `https://git.mimuw.edu.pl/IPP-login.git` (gdzie login to identyfikator używany do logowania się w laboratorium komputerowym) w gałęzi `template/part2` umieścimy testy. Zastrzegamy sobie możliwość wgrania poprawek. Lista poprawek:

- na razie nie ma poprawek.

Wymagamy

Jako rozwiązanie części 2 zadania wymagamy:

- umieszczenia kodu źródłowego implementacji w katalogu `src`,
- uzupełnienia dokumentacji dla programu `doxygen`,
- dostosowania pliku konfiguracyjnego dla programu `cmake`,
- stworzenia skryptu o nazwie `chain_poly.sh` i umieszczenia go w głównym katalogu rozwiązania.

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 `calc_poly` oraz dokumentacja. W poleceniu `cmake` powinno być również możliwe jawne określenie wariantu budowania pliku wynikowego:

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

Pliki z rozwiązania części 1 i dostarczone przez nas pliki można modyfikować, o ile nie zmienia to działania programu i zachowuje wymagania podane w treści zadania, przy czym nie należy zmieniać opcji kompilacji.

Wymagania dotyczące implementacji

Złożoność

Złożoność operacji na wielomianach wynika ze złożoności z części 1 zadania. Program implementuje maszyną stosową. Operacje stosowe powinny działać w czasie stałym.

Limity czasowe użyte przy testowaniu:

- 60s dla `test_cmds_one_var`, `test_cmds_many_var{1,2,3}`, `test_deep_stack`, `test_many_rows`, `test_long_row_one_var`, `test_long_row_many_var`,
- 1s dla pozostałych testów.

Inne

Niepowodzenie alokacji pamięci należy wykrywać za pomocą asercji.

Oddawanie rozwiązania

Rozwiązanie należy oddawać, podobnie jak części 1, przez repozytorium git. W repozytorium mają się znaleźć wszystkie pliki niezbędne do zbudowania pliku wykonywalnego `calc_poly` i dokumentacji oraz skrypt `chain_poly.sh`. *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 2 zadania wielomiany znajduje się w repozytorium w commicie 518507a7e9ea50e099b33cb6ca3d3141bc1d6638.

Rozwiązanie należy zatwierdzić (git commit) i wysłać do repozytorium (git push) najpóźniej do godz. 6.00 dnia 27 maja 2017 (sobota). Dopuszczamy także wysyłanie po tym terminie z karą 2 pkt. Jednakże ostateczny termin wysyłania jest do godz. 8.00 dnia 28 maja 2017 (niedziela).

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 program nie przejdzie, traci się 1 punkt;
- za złe działanie skryptu można stracić do 5 punktów;
- za wycieki pamięci można stracić do 6 punktów;
- za niezgodną ze specyfikacją strukturę katalogów i 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 | Nie próbowano |
| Stan oceniania | Nie ocenione |
| Termin oddania | sobota, 27 maj 2017, 06:00 |
| Pozostały czas | 1 dzień 7 godz. |
| Ostatnio modyfikowane | - |
| Komentarz do przesłanego zadania | ► Komentarze (0) |

Dodaj swoją pracę

Dodaj lub edytuj swoje zadanie

NAWIGACJA



Kokpit

- Strona główna

Strony


Moje kursy

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

Temat 9

 **Wielomiany, część 2**

 wielomiany2-tests-v2.zip

 Bash-extra

Temat 10

Temat 11

WPI.INFO.I.16/17

WPI.LAB.INFO.I.16/17

PO.INFO.I.16/17

MD.INFO.I.16/17

ADMINISTRACJA



Administracja kursem

