Wrocław 2023.10.01

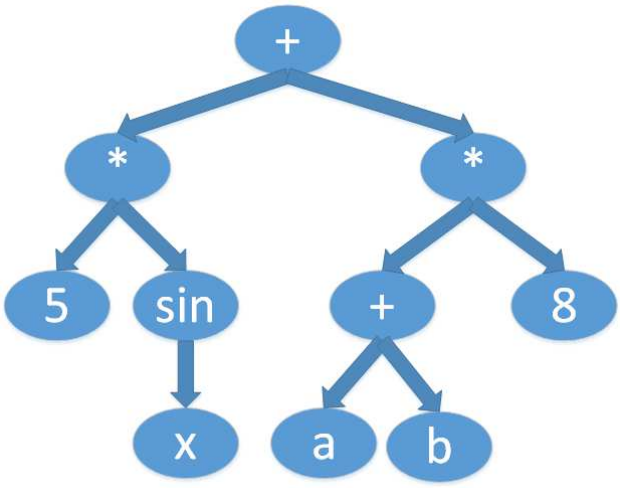
Autor: Michał Przewoźniczek

Techniki Efektywnego Programowania – zadanie 3
Zależności pomiędzy klasami i obiektami, obsługa drzew

Zadanie
UWAGI:

1. **Pisząc własny program można użyć innego nazewnictwa niż to przedstawione w treści zadania i w przykładach. Należy jednak użyć jakiejś spójnej konwencji kodowania, zgodnie z wymaganiami kursu.**
2. **Nie wolno używać wyjątków (jest to jedynie przypomnienie, wynika to wprost z zasad kursu).**
3. **Wolno używać wyłącznie komend ze standardu C++98**
4. **Interfejs użytkownika ma być również wykonany obiektowo, zgodnie z regulaminem kursu.**

Program ma umożliwiać wczytywanie formuł matematycznych, interpretowanie ich w postaci drzewa i wykorzystanie drzewa do wyliczenia wartości formuły. Na przykład formułę: 5*sin(x) + (a+b)*8 można zinterpretować jako drzewo przedstawione na Rys. 1.



Rys. 1 Przykładowe drzewo do przetwarzania formuł matematycznych

Żeby wyliczyć wartość formuły z powyższego przykładu, program musi znać wartości przyjmowane przez zmienne *x*, *a* oraz *b*. Należy zauważyć, że jeśli na podstawie dowolnej formuły, prawidłowo skonstruowano drzewo, wtedy istnieje możliwość wyliczenia wartości formuły, dla dowolnych wartości zmiennych, które mogą występować w drzewie.

Dla uproszczenia zadania wczytywania formuł matematycznych, będą podawane na wejście programu w postaci prefiksowej (czyli w tzw. Notacji Polskiej). Notacja prefiksowa tym różni się od notacji infiksowej (używanej na co dzień), że najpierw podaje się operator, a dopiero potem

zmienne. W notacji prefiksowej nie jest potrzebne używanie nawiasów. Formuła 5*sin(x) + (a+b)*8 zostanie w postaci prefiksowej zapisana jako: + * 5 sin x * + a b 8.

Komunikacja z programem ma odbywać się za pośrednictwem konsoli.

Polecenia, które można wprowadzić do konsoli

enter <formula> - wykonanie powoduje próbę stworzenia drzewa na podstawie podanego wyrażenia. Jeżeli wyrażenie nie może zostać wczytane, bo posiada błędy, program ma uzupełnić drzewo tak, żeby drzewo było prawidłowe, wypisać odpowiedni komunikat, oraz poinformować, jaka formuła ostatecznie będzie przetwarzana (należy ją wypisać na ekranie). Naprawa wyrażenia ma nastąpić poprzez jego uzupełnienie. Dokładne wytyczne znajdują się w punkcie 1 na stronie 3.

vars – wypisuje wszystkie zmienne z aktualnie wprowadzonego drzewa. Każda zmienna jest wypisywana jeden raz, nawet jeśli występuje w drzewie więcej razy.

print – wypisuje aktualnie wprowadzone drzewo w postaci prefiksowej

comp <var0> <var1>…<varN> – wyliczenie wartości wprowadzonej formuły, gdzie *<var0> <var1>…<varN>* to wartości dla kolejnych zmiennych występujących w drzewie. Wartości odpowiadają zmiennym w kolejności w jakiej zmienne są wypisywane przez polecenie *vars*. Na przykład, jeśli *vars* wypisało 3 zmienne: a b x, to polecenie „*comp 1 2 3*” wyliczy wartość drzewa dla a=1, b=2 i x=3. Jeśli liczba podanych wartości *<var0> <var1>…<varN>* nie będzie odpowiadać liczbie zmiennych w drzewie program ma wypisać odpowiedni komunikat.

join <formula> - wykonanie powoduje próbę stworzenia drzewa na podstawie podanego wyrażenia. Następnie stworzone drzewo jest doklejane do istniejącego drzewa. Patrz punkt 2 na stronie 3. Do wykonania tej komendy należy użyć operatorów „+” i „=”, zgodnie z opisem podanym w wymaganiach dotyczących implementacji.

Przykład:

enter + a (próba wczytania nowego wyrażenia; występuje błąd, bo brakuje jednego argumentu; program ma wypisać komunikat o błędnym wyrażeniu i je poprawić, na przykład poprzez dodanie drugiego argumentu dodawania z domyślną wartością 1, wtedy program informuje, że przetwarzana będzie formuła „+ a 1”)
enter + 3 (próba wczytania nowego wyrażenia; występuje błąd, bo jest za mało argumentów; program ma wypisać komunikat o błędnym wyrażeniu i je poprawić poprzez uzupełnianie, a następnie poinformować użytkownika, że przetwarzana będzie np. formuła „+ 3 1”)
enter 3 20 30 (próba wczytania nowego wyrażenia; występuje błąd, bo jest za dużo argumentów; program ma wypisać komunikat o błędnym wyrażeniu i je poprawić. W tym przypadku pierwszy symbol „3” stanowi jednowęzłowe, prawidłowe drzewo, pozostałe symbole powinny zostać zignorowane, o czym należy poinformować użytkownika)

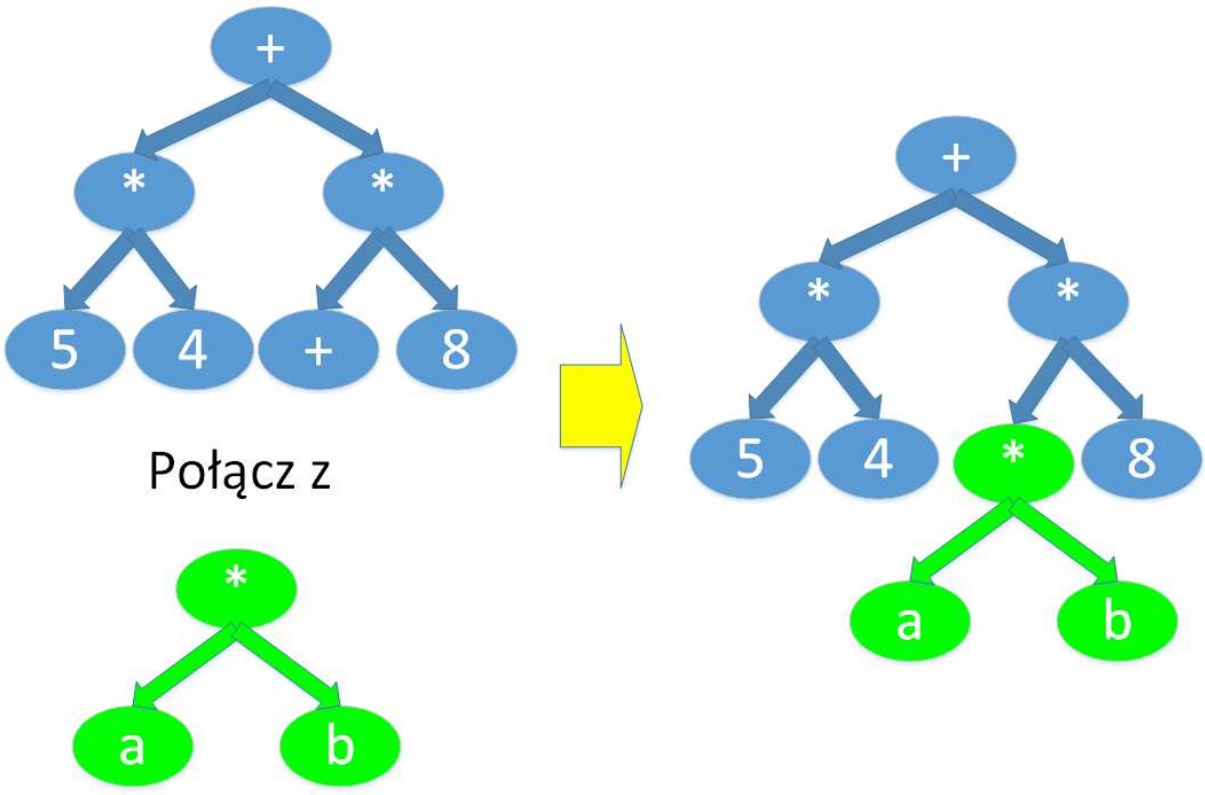
enter + a b (próba wczytania nowego wyrażenia; brak błędów)

print (wypisanie drzewa w postaci prefiksowej, czyli „+ a b”)
comp 1 2 3 (próba wyliczenia wartości formuły – nieudana, bo wartości jest więcej niż zmiennych; program wyświetla odpowiedni komunikat)
comp 1 2 (próba wyliczenia wartości formuły dla a = 1 i b = 2; program wyświetla odpowiedni komunikat, że wynik to 3)
*join * 2 1* (próba połączenia wprowadzonej formuły z formułą „* 2 1”; brak błędów; w wyniku dostajemy drzewo „+ a * 2 1”, lub „+ * 2 1 b”)

Pozostałe wymagania do działania programu

1. Poprawianie wyrażeń poprzez uzupełnianie brakujących elementów. Na przykład, jeśli zostanie wprowadzone wyrażenie „+ + +”, to takie wyrażenie może zostać naprawione na przykład tak: „ + **1** + **1** + **1** **1**”. W tej sytuacji pierwotne, niekompletne wyrażenie zostało uzupełnione innymi symbolami. Nie jest dopuszczalne, żeby w ramach naprawy otrzymano wyrażenie „+ 1 1”, bo wtedy dwa, z trzech działań „+” zostaną pominięte. Jeżeli wprowadzone zostanie wyrażenie „+ 1 1 2”, to naprawiając takie wyrażenie należy otrzymać „+ 1 1”. „2” może zostać pominięte, ponieważ wcześniej prawidłowo wczytano kompletne wyrażenie.

2. Łączenie drzew. Jeżeli łączymy drzewo A z drzewem B, to znaczy, że zamiast jednego z liści drzewa A (może być wybrany dowolny liść) wstawiamy korzeń drzewa B. Operacja ta jest pokazana na Rys. 2.



Rys. 2 Operacja łączenia drzew

3. Należy przyjąć, że poszczególne człony wyrażenia są oddzielane od siebie co najmniej jedną spacją.

4. Wyrażenie może zawierać symbole operacji, zmienne i liczby

   - Wymagane typy operacji: +, -, *, /, sin, cos

   - Nazwy zmiennych

     - składają się z liter (dużych i małych) i cyfr (jeśli w ciągu znaków znajduje się jakaś niedozwolona wartość np. „$”, to można ją zignorować i poinformować o tym użytkownika)
     - liter i cyfr może być dowolnie dużo
     - w nazwie zmiennej musi być co najmniej jedna litera

   - Liczby składają się wyłącznie z cyfr i mogą być jedynie dodatnie

Wymagania dotyczące implementacji

W warstwie obliczeniowej należy użyć co najmniej dwóch klas: CTree, oraz CNode. Pierwsza obsługuje drzewo w ogólności i to z nią komunikuje się warstwa interfejsu. Druga klasa służy do obsługi pojedynczego węzła w drzewie. Należy pamiętać, że drzewo wynikające z wczytania wyrażenia w ogólnym przypadku nie jest drzewem binarnym, ponieważ można definiować funkcje pobierające więcej niż dwa argumenty.

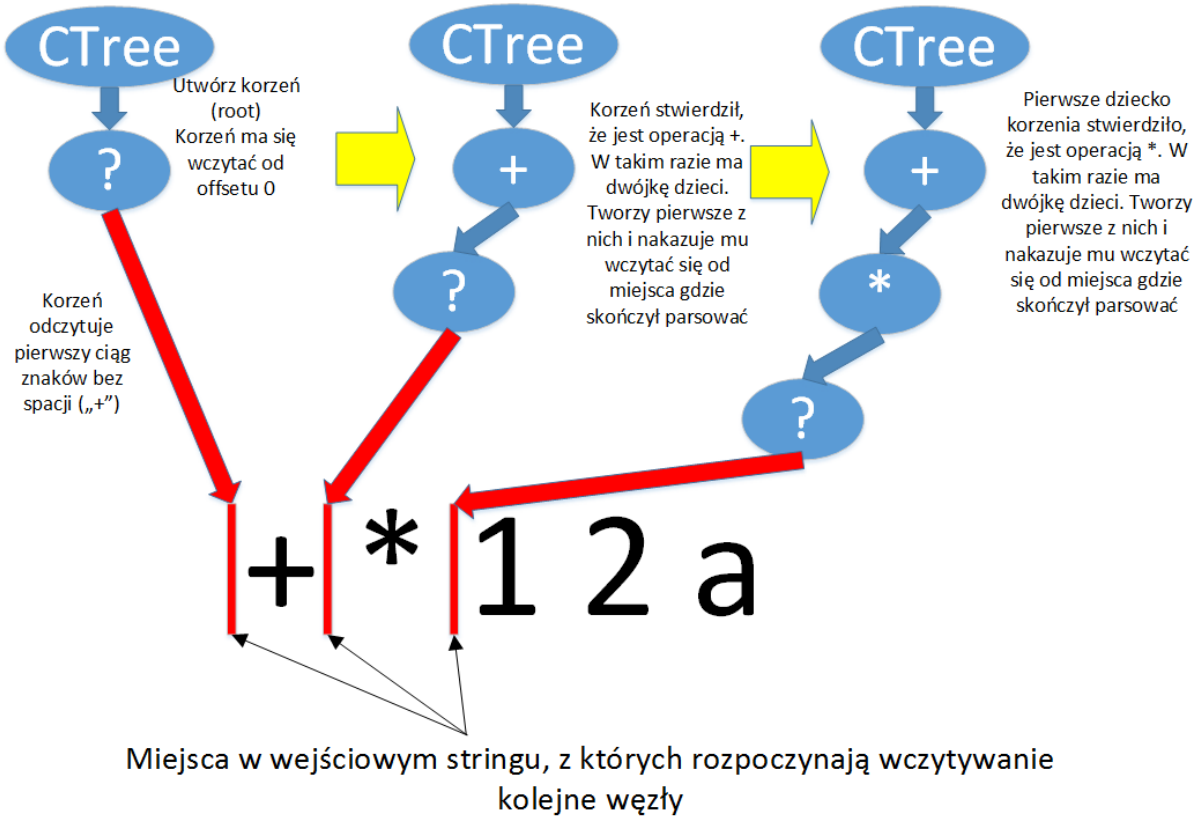
Klasa ma posiadać oprogramowane operatory = i +. Tak, aby możliwe było wykonanie kodu:

```
CTree c_sum, c_a, c_b;
//inicjacja obiektów
c_sum = c_a + c_b;
```

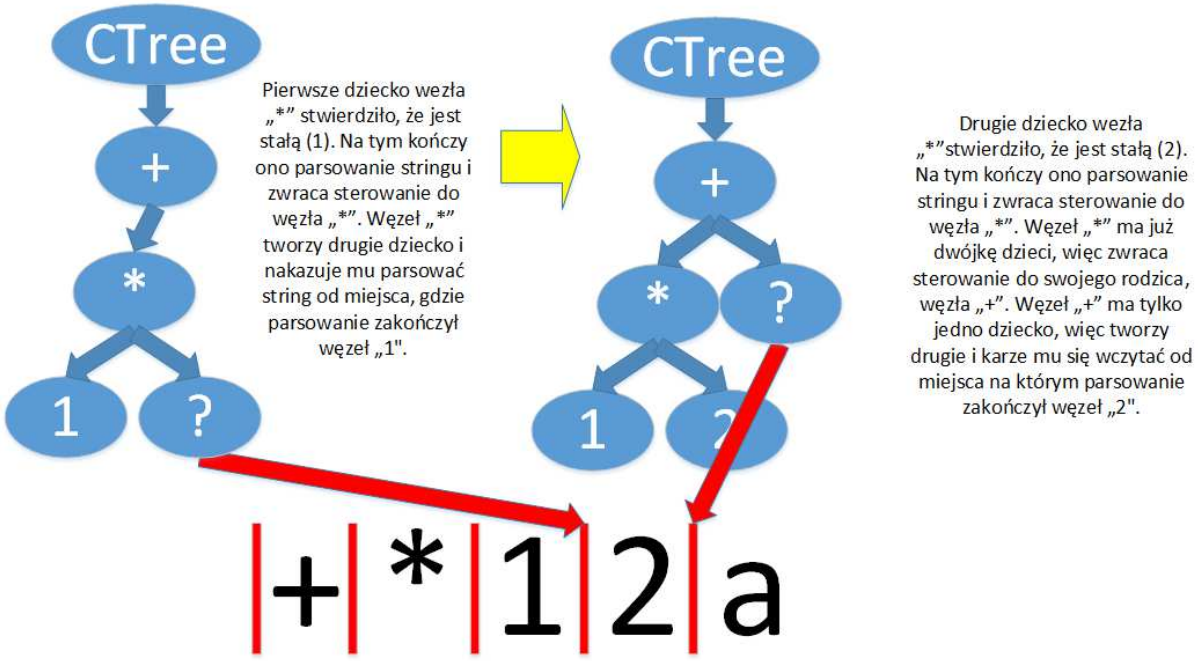
Po wykonaniu ostatniej instrukcji stan obiektów c_a i c_b pozostaje niezmieniony, a w c_sum znajduje drzewo będące połączeniem drzew z c_a i c_b, zgodnie z opisem przedstawionym w punkcie 2 na stronie 3. Należy zauważyć, że takie dodawanie nie jest przemienne.

Wymagania dotyczące wczytywania wyrażenia

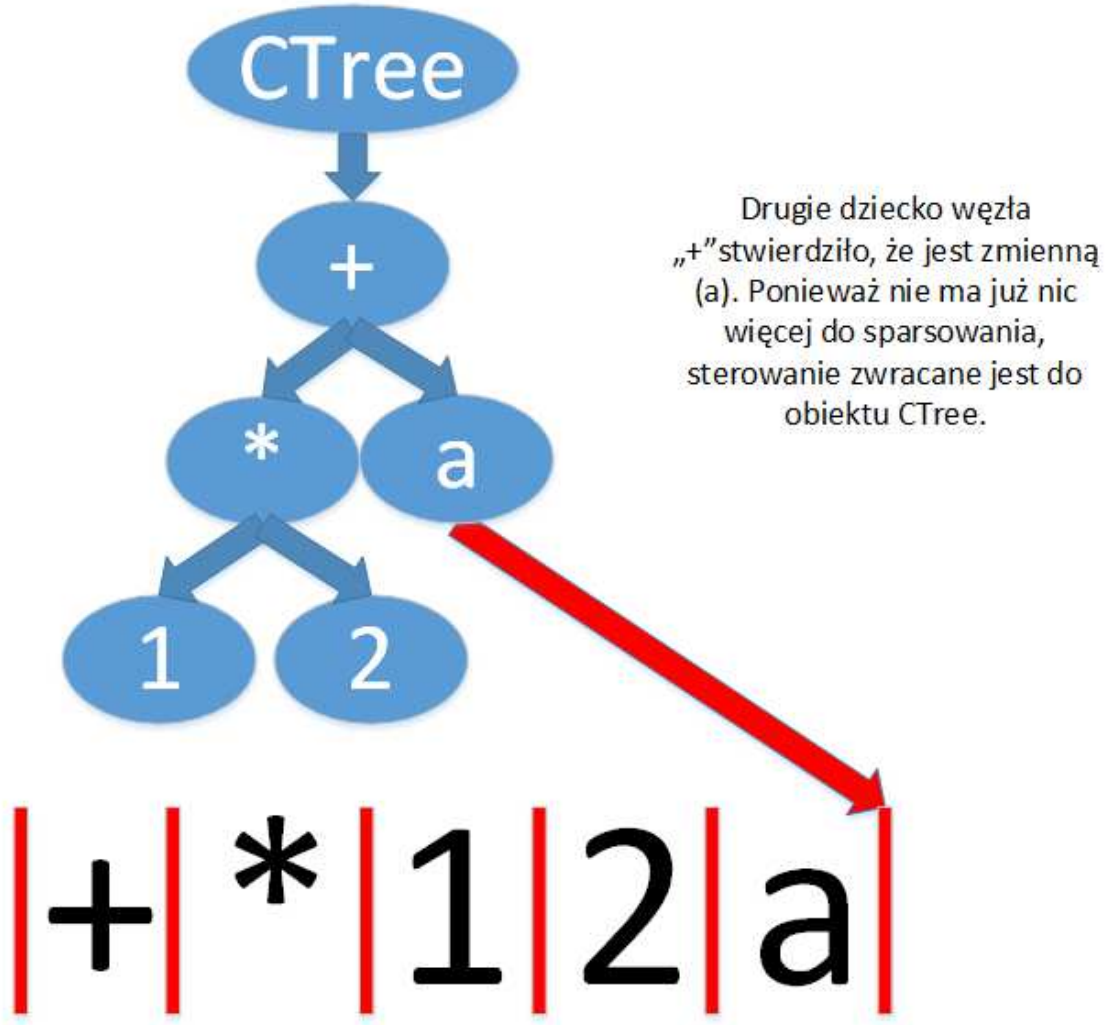
Operacja wczytywania wyrażenia ma zostać zaimplementowana obiektowo, w sposób opisany poniżej. W szczególności zakazane jest wczytywanie wyrażenia za pośrednictwem stosu. Wymagany sposób wczytywania danych ze zmiennej typu string został przedstawiony na rysunku 2, 3 i 4.



Rys. 3 Operacja wczytywania drzewa ze zmiennej typu string część 1



Rys. 4 Operacja wczytywania drzewa ze zmiennej typu string część 2



Rys. 5 Operacja wczytywania drzewa ze zmiennej typu string część 3

Zalecana literatura

Wykład

Materiały możliwe do znalezienia w Internecie