Metody Translacji

Zadanie zaliczeniowe - rok 2019/20

Poniżej opisane są rozszerzenia języka Mini, których implementacja wymagana jest dla uzyskania oceny 4.5 i 5. Rozszerzony język nazwijmy Mini++.

Opis języka Mini++

1) Elementy języka.
W języku Mini++ dochodzą następujące terminale nie występujące w języku Mini.
- słowa kluczowe: **create break continue**
- operatory i symbole specjalne: **[ ] ,**

2) Zmiany w deklaracjach
W języku Mini++ można w jednej deklaracji deklarować kilka zmiennych, czyli pojedyncza deklaracja składa się z typu, listy identyfikatorów rozdzielonych przecinkami i średnika.

3) Zmiany w instrukcji blokowej
W języku Mini++ deklaracje mogą wystąpić w każdym bloku (w języku Mini były dozwolone jedynie w głównym bloku programu). Innymi słowy W języku Mini++ znika rozróżnienie składniowe pomiędzy głównym blokiem programu, a blokami wewnętrznymi. W Mini++ każdy blok to ujęte w nawiasy klamrowe ciąg deklaracji i ciąg instrukcji (oczywiście każdy z tych ciągów może być pusty). W blokach wewnętrznych można deklarować zmienne o takich samych nazwach jak zmienne zadeklarowane w blokach zewnętrznych i nie powoduje to konfliktów. W takim przypadku zmienne z bloków zewnętrznych są przesłaniane przez zmienne z bloków wewnętrznych i są w bloku wewnętrznym niedostępne, ale nadal istnieją i są znowu dostępne po zakończeniu bloku wewnętrznego.

Uwaga: Zmienne zadeklarowane w blokach wewnętrznych są niezainicjowane (tzn. ich początkowe wartości są nieznane i mogą być dowolne), to różnica w stosunku do zmiennych zadeklarowanych w bloku głównym, które są niejawnie inicjowane wartościami zerowymi (dla zachowania zgodności z językiem Mini).

Przykład:

```
program
    {
    double x;
    ...
    x = 2.5;
    ...
    while (...)
        {
        int x;  // od teraz w tym bloku zmienna x
        ...     // z bloku zewnętrznego jest niedostępna
        x = 1;
        ...
        }       // tu "likwidowana" jest zmienna z bloku wewnętrznego
    ...
    write x;    // zmienna z bloku zewnętrznego jest znowu dostępna
    ...
    }
```

4) Instrukcja break
Istnieją dwie postacie instrukcji break

A) Postać klasyczna to po prostu słowo kluczowe break, po którym następuje średnik.
Instrukcja break w tej postaci musi być umieszczona wewnątrz pętli (w przeciwnym przypadku należy zgłosić błąd kompilacji), a jej działanie jest dobrze znane - przerywa pętlę wewnątrz której jest umieszczona.

B) Postać rozszerzona to słowo kluczowe break, po którym następuje stała całkowita, a po niej średnik. Podana stała określa wewnątrz ilu zagnieżdżonych pętli musi być umieszczona ta postać instrukcji break (jeśli tak nie jest należy zgłosić błąd kompilacji). Działanie takiej instrukcji break polega na przerwaniu wskazanej przez stałą liczby zagnieżdżonych pętli (np. instrukcja break 2 przerywa 2 zagnieżdżone pętle).

Uwaga 1: Instrukcja break 1 jest równoważna klasycznej instrukcji break
Uwaga 2: Stała musi być dodatnia (instrukcja break 0 jest niepoprawna, należy zgłosić błąd kompilacji).

5) Instrukcja continue
Instrukcja continue to po prostu słowo kluczowe continue, po którym następuje średnik.
Instrukcja continue musi być umieszczona wewnątrz pętli (w przeciwnym przypadku należy zgłosić błąd kompilacji), a jej działanie jest dobrze znane - powoduje zakończenie bieżącej iteracji i przejście do sprawdzenia warunku zakończenia pętli.

6) Tablice
To największe rozszerzenie, składa się ono z kilku elementów.

A) Deklaracje tablic
Deklaracja tablic składa się z następujących elementów (występujących w podanej tu kolejności):
- typ elementów tablicy
- deklaracja liczby wymiarów tablicy
- lista rozdzielonych przecinkami identyfikatorów
- średnik
Dozwolone typy elementów tablic są takie same jak dozwolone typy zmiennych prostych.
Deklaracja liczby wymiarów tablicy to ujęta w nawiasy kwadratowe lista przecinków. Pusta lista przecinków oznacza tablicę jednowymiarową, jednoelementowa lista przecinków oznacza tablicę 2-wymiarową, ogólnie n-elementowa lista przecinków oznacza tablicę n+1 wymiarową.
Lista identyfikatorów to lista nazw deklarowanych tablic.

Przykład

```
double[] tabd1; // deklaracja 1-wymiarowej tablicy o elementach typu double
int[,,]  tabi3; // deklaracja 3-wymiarowej tablicy o elementach typu int
int[]    tx, ty; // deklaracja dwóch 1-wym. tablic o elementach typu int
```

Uwaga 1: Jak widać składnia deklaracji tablic w Mini ++ jest analogiczna (identyczna) do deklaracji tablic wielowymiarowych w C#.
Uwaga 2: W deklaracji tablic deklarujemy liczbę wymiarów tablicy - nie deklarujemy liczby elementów w poszczególnych wymiarach (czyli nie deklarujemy rozmiaru tablicy).

B) Tworzenie tablic
Do tworzenia tablic służy instrukcja create, która składa się z następujących elementów:
słowa kluczowego create, opisu tablicy i średnika.
Opis tablicy to identyfikator, po którym następuje ujęta w nawiasy kwadratowe lista wyrażeń określających liczbę elementów w poszczególnych wymiarach tablicy. Wyrażenia rozdzielone są przecinkami.
Instrukcja create jest odpowiednikiem operatora new z języka C#, w szczególności
- elementami wyrażeń określających rozmiar tablicy mogą być zmienne
- dla tego samego identyfikatora zadeklarowanego jako tablica można wielokrotnie wykonać instrukcję create
- liczba wymiarów w deklaracji zmiennej tablicowej i instrukcji create musi być jednakowa
(i wszystkie muszą być określone - nie wolno tworzyć tablicy "po kawałku")

Przykład

Dla deklaracji
```
int[,] tab;
```

poprawna jest instrukcja
```
create tab[n*m+2,k-1];
```

błędne są następujące instrukcje (powinny być sygnalizowane jako błędy kompilacji)
```
create tab[k-1];  // niezgodna liczba wymiarów
create tab[n,];   // brak określenia liczby elementów w drugim wymiarze
```

natomiast instrukcja
```
create tab[n,-1];
```
jest poprawna składniowo (nie powinna generować błędów kompilacji), natomiast powinna spowodować błąd wykonania (ale nie należy się tym zajmować - .NET się tym zajmie).

C) Odwołania do elementów tablic
Odwołania do elementów tablic są składniowo identyczne jak w języku C# i składają się z identyfikatora, po którym następuje ujęta w nawiasy kwadratowe lista wyrażeń określających indeksy w poszczególnych wymiarach tablicy. Wyrażenia rozdzielone są przecinkami.

Przykład

Podobnie jak w języku Mini deklaracje nie zawierają inicjalizatorów, ale zmienne inicjowane są niejawnie wartościami zerowymi odpowiedniego typu.

Dla deklaracji
```
int[,] tab;
```

poprawne są odwołania
```
n = tab[i,j+1];
tab[0,k] = 5;
```

błędne są odwołania (powinny być sygnalizowane jako błędy kompilacji)
```
tab1 = tab[k-1];
tab[k-1] = tab1;
```

natomiast nie należy zajmować się problemem przekroczenia zakresu przez indeks - to powinno spowodować błąd wykonania, ale zajmie się tym .NET).