Politechnika Warszawska

Informatyka Stosowana Wydział Elektryczny
Języki formalne i kompilatory

Język A
Własny język programowania

Przygotowane przez: Bartłomiej Przygodzki
Prowadzący: Chaber Bartosz
Data: 1 czerwca 2022

Spis treści

1 Elementy leksykalne

1.1 Identyfikatory

Identyfikatory to sekwencje znaków używanych do nazywania zmiennych i funkcji.

Identyfikator może składać się jedynie z podstawowych znaków z alfabetu łacińskiego. Nie może być w nim liczb ani znaków specjalnych.

Znaki duże i małe są rozróżnialne.

1.2 Słowa kluczowe

Słowa kluczowe to specjalne identyfikatory zarezerwowane dla samego języka programowania. Nie można użyć ich jako własne identyfikatory.

Lista słów kluczowych:

string int real if else loop def return read print

1.3 Operatory

Operator to specjalny token, który wykonuje operację, taką jak dodawanie lub odejmowanie.

1.3.1 Operator przypisania

Operator przypisania '=' służy do przechowania wartości w zmiennej.

```
int x = 9
```

1.3.2 Operatory arytmetyczne

Język A zapewnia operatory do standardowych operacji arytmetycznych: dodawania, odejmowania, mnożenia i dzielenia. Operacje te możliwe są do wykorzystania ze stałymi, zmiennymi jak i funkcjami. Możliwe jest wykonywanie wielu operacji arytmetycznych w jednej linijce kodu.

```
int a = 1 + b * fun()
```

Należy zadbać aby wszystkie elementy operacji arytmetycznych dzieliły ten sam typ.

1.3.3 Operatory porównania

Operatory porównania służą do decydowania o wykonaniu instrukcji if. Wyróżniamy:

== operator porównania decyduje o wykonaniu instrukcji if kiedy identyfikator jest równy podanej liczbie

!= operator porównania decyduje o wykonaniu instrukcji if kiedy identyfikator nie jest równy podanej liczbie

> operator porównania decyduje o wykonaniu instrukcji if kiedy identyfikator jest większy od podanej liczby

< operator porównania decyduje o wykonaniu instrukcji if kiedy identyfikator jest mniejszy od podanej liczby

```
if(a!=1)
{
```

```
a=1
}
```

1.3.4 Rzutowanie typów

Możliwe jest użycie rzutowania typu, aby jawnie spowodować, że wyrażenie będzie mieć określony typ danych. Dostępne są rzutowania na typ int oraz real. Nazwa typu na który rzutujemy musi być otoczona nawiasami.

```
real test = (real)(fun()) + (real)(a)
```

Jest to bardzo przydatne aby umożliwić zgodność typów. Typy mogą być zmieniane tylko ręcznie.

1.4 Separatory

Separator oddziela instrukcje. Jest nim znak nowej linii. Na końcu każdej instrukcji musi znajdować się znak nowej linii.

1.5 Biała przestrzeń

Białe znaki są ignorowane. Są nimi spacje i tabulacja. W stałych łańcuchowych spacje i tabulatory nie są ignorowane.

1.6 Komentarze

Komentarz rozpoczyna się znakiem # a kończy nową linią.

```
int y = 1 # to jest komentarz
```

2 Typy

2.1 Typ całkowity

Język A posiada 32 bitowy typ int. Umożliwia on przechowanie liczby od -2,147,483,648 do 2,147,483,647.Poniżej widnieje przykład deklaracji zmiennej typu int.

```
int a
a =1
int b = 2
```

Możliwa jest sama deklaracja zmiennej bez przypisywania jej liczby. Wtedy kompilator automatycznie przypisuje jej wartość 0.

2.2 Typ rzeczywisty

Istnieje jeden typ liczb ułamkowych i jest nim real. Jest on 62 bitowy. Poniżej widnieje przykład deklaracji zmiennej typu real.

```
real a
a =1
real b = 2
```

Możliwa jest sama deklaracja zmiennej bez przypisywania jej liczby. Wtedy kompilator automatycznie przypisuje jej wartość 0,0.

2.3 Ciąg znaków

String to typ zarezerwowany do zapisywania ciągu znaków. Za jego pomocą można zapamiętać ciąg za pomocą identyfikatora a następnie podać go do funkcji print. Poniżej widnieje sposób użycia.

```
string tekst ="World"
print "Hello "
print tekst
```

Nie istnieje możliwość modyfikowania stringa. Możliwe jest jedynie jego nadpisanie. Nie można wykorzystywać go do obliczeń arytmetycznych.

3 Instrukcje

3.1 Wyrażenia

Operacje przypisania z możliwością zastosowania operatorów są wyrażeniem. Niemożliwe jest zapisanie pojedynczej stałej lub równania bez przypisania jako wyrażenie.

```
5
a + 2
10 >= b
```

Program dla powyższych przykładów zwróci błąd.

3.2 Instrukcja if

Możesz użyć instrukcji if do warunkowego wykonania części programu na podstawie wartości logicznej danego wyrażenia.

```
if (wyrażenie){
blok A
}else{
blok B
}
```

Instrukcja w bloku A wykona się kiedy wyrażenie będzie prawdziwe. Wyrażenia zostały opisane w rozdziale 1.3.2. Kiedy wyrażenie będzie niepoprawne wykona się blok B. Część else można pominąć w takim przypadku żadna instrukcja nie zostanie wykonana w przypadku niepowodzenia wyrażenia. W blokach mogą być zapisywane wszystkie instrukcje oprócz deklaracji funkcji. Możliwe jest zapisanie if-a wewnątrz innego if-a.

3.3 Instrukcja loop

Instrukcja loop służy do powtarzania całkowitą ilość razy danego fragmentu kodu. Schemat zastosowania podany jest poniżej.

```
loop iterator
{
instrukcje
}
```

Interatorem może być tylko liczba całkowita jak np 5 lub zmienna o typie całkowitym.

3.4 Instrukcja return

Jest to obowiązkowa instrukcja na końcu każde funkcji. Musi ona zwracać liczbę całkowitą. Może to być stała lub zmienna.

```
return a
lub
return 0
```

4 Funkcje

4.1 Definicja funkcji

Aby zadeklarować funkcje należy użyć poniższego schematu.

```
def funid
{
instrukcje
return
}
```

Gdzie funid trzyma się takich samych zasad jak zwykłe id. Deklaracja funkcji nie może zawierać w sobie jedynie drugiej deklaracji funkcji. Funkcja zawsze zwraca typ int. Funkca nie może przyjmować żadnych parametrów. Ma ona własną przestrzeń zmiennych lokalnych. Może ona korzystać ze zmiennych globalnych, choć najpierw wyszukuje id ze zmiennych lokalnych.

4.2 Odwołanie do funkcji

Do funkcji odwołujemy się poprzez wyrażenie.

```
nazwa_funkcji()
np
fun()
```

Funkcje mogą być wykorzystywane do obliczeń arytmetycznych gdyż zawsze zwracają zmienna int.

5 Operacje IO

5.1 Funkcja print

Funkcja print wyświetla na standardowym wejściu takie zmienne jak string int oraz real. Wyświetlany parametr może być podany jako zmienna lub stała.

```
string tekst ="World"
print "Hello "
print tekst
print 8.5
a=2
print a
```

5.2 Funkcja read

Funkcja read sczytuje ze standardowego wejścia daną i zapisuje ją w pamięci programu. W przypadku egzekucji programu czeka ona na wejście z konsoli. W celu akceptacji wpisywanego wejścia należy zakończyć go znakiem enter. Wpisywana zmienna może być typu int lub real.

```
int a
read a
```

Zmienna podana do funkcji read musi być wcześniej zadeklarowana.

6 Przykładowy program

Przykładowy program który podaje wartość danego wyrazu ciągu Fibonacciego.

```
print "Wpisz iterator calkowity "
int iterator
read iterator

int x = 0
int y = 1

def fibonacci{
int wynik

if (iterator ==0){
    wynik =0
}else{
    if (iterator ==1){
        wynik =1
    }else{
        if (iterator ==2){
                wynik =1
        }else{
            iterator=iterator - 1
            loop iterator{
            wynik = x + y
            x = y
            y = wynik
            }
            }
    }
}


return wynik
}


int wynik = fibonacci()

print "Wartosc ciagu Fibonacciego o wpisanym wyrazie wynosi   "
print wynik
```