



FAKULTA APLIKOVANÝCH VĚD  
ZÁPADOČESKÉ UNIVERZITY  
V PLZNI

KATEDRA INFORMATIKY  
A VÝPOČETNÍ TECHNIKY



**Semestrální práce**

## **Překladač jazyka Ligma**

Milan Janoch a Jakub Pavlíček



# Obsah

<b>1</b>	<b>Zadání</b>	<b>2</b>
<b>2</b>	<b>Návrh jazyka</b>	<b>5</b>
2.1	Zvolená rozšíření jazyka Ligma . . . . .	5
2.2	Omezení jazyka . . . . .	5
2.3	Konstrukce jazyka . . . . .	6
2.3.1	Povinná . . . . .	6
2.3.2	Rozšiřující . . . . .	7
<b>3</b>	<b>Testování</b>	<b>10</b>
<b>4</b>	<b>Závěr</b>	<b>11</b>
	<b>Seznam výpisů</b>	<b>12</b>

# Zadání

## 1

Cílem práce bude vytvoření překladače zvoleného jazyka. Je možné inspirovat se jazykem PL/0, vybrat si podmnožinu nějakého existujícího jazyka nebo si navrhnout jazyk zcela vlastní. Dále je také potřeba zvolit si pro jakou architekturu bude jazyk překládán (doporučeny jsou instrukce PL/0, ale je možné zvolit jakoukoliv instrukční sadu pro kterou budete mít interpret).

Jazyk musí mít minimálně následující konstrukce:

- definice celočíselných proměnných
- definice celočíselných konstant
- přiřazení
- základní aritmetiku a logiku (+, -, \*, /, AND, OR, negace a závorky, operátory pro porovnání čísel)
- cyklus (libovolný)
- jednoduchou podmínku (if bez else)
- definice podprogramu (procedura, funkce, metoda) a jeho volání

Překladač který bude umět tyto základní věci bude hodnocen deseti body. Další body (alespoň do minimálních 20) je možné získat na základě rozšíření, jsou rozděleny do dvou skupin, jednodušší za jeden bod a složitější za dva až tři body. Další rozšíření je možno doplnit po konzultaci, s ohodnocením podle odhadnuté náročnosti.

Jednoduchá rozšíření (1 bod):

- každý další typ cyklu (for, do .. while, while .. do, repeat .. until, foreach pro pole)

- else větev
- datový typ boolean a logické operace s ním
- datový typ real (s celočíselnými instrukcemi)
- datový typ string (s operátory pro spojování řetězců)
- rozvětvená podmínka (switch, case)
- násobné přiřazení ( $a = b = c = d = 3;$ )
- podmíněné přiřazení / ternární operátor ( $min = (a < b) ? a : b;$ )
- paralelní přiřazení ( $a, b, c, d = 1, 2, 3, 4;$ )
- příkazy pro vstup a výstup (read, write - potřebuje vhodné instrukce které bude možné využít)

Složitější rozšíření (2 body):

- příkaz GOTO (pozor na vzdálené skoky)
- datový typ ratio (s celočíselnými instrukcemi)
- složený datový typ (Record)
- pole a práce s jeho prvky
- operátor pro porovnání řetězců
- parametry předávané hodnotou
- návratová hodnota podprogramu
- objekty bez polymorfismu
- anonymní vnitřní funkce (lambda výrazy)

Rozšíření vyžadující složitější instrukční sadu než má PL/0 (3 body):

- dynamicky přiřazovaná paměť - práce s ukazateli
- parametry předávané odkazem
- objektové konstrukce s polymorfním chováním
- instanceof operátor
- anonymní vnitřní funkce (lambda výrazy) které lze předat jako parametr

- mechanismus zpracování výjimek

Vlastní interpret (řádkový, bez rozhraní, složitý alespoň jako rozšířená PL/0) je za 6 bodů.

Kromě toho že by program měl fungovat se zohledňují i další věci, které mohou pozitivně nebo negativně ovlivnit bodování:

- testování - tvorba rozumné automatické testovací sady +3 body (pro inspiraci hledejte test suit pro LLVM nebo se podívejte na Plum Hall testy, ale jde skutečně jen o inspiraci, stačí výrazně jednodušší řešení).
- Kvalita dokumentace -x bodů až +2 body podle kvality a prohrěšků (vynechaná gramatika, nesrozumitelné věty, příliš chyb a překlepů, bitmapové obrázky pro diagramy s kompresními artefakty, ...).
- Vedení projektu v GITu -x bodů až +2 body podle důslednosti a struktury příspěvků.
- Kvalita zdrojového textu -x bodů až +2 body podle obecně známých pravidel ze ZSWI, PPA a podobně (magická čísla, struktura programu a dekompozice problému, božské třídy a metody, ...)

# Návrh jazyka

## 2

### 2.1 Zvolená rozšíření jazyka Ligma

- cyklus do-while
- cyklus for
- cyklus repeat-until
- datový typ boolean a logické operace s ním
- else větev
- násobné přiřazení ( $a = b = c = d = 3;$ )
- parametry předávané hodnotou
- návratová hodnota podprogramu

### 2.2 Omezení jazyka

- Při deklaraci proměnné je třeba vždy nastavit hodnotu
- Funkce musí být definovány v nejvrchnějším scope
- Funkce podporují pouze návratový typ int
- V hlavičce for cyklu se musí deklarovat proměnná
- Podporované datové typy proměnných: int, boolean
- Podporovaná aritmetika pro boolean - AND, OR, negace, ==, !=
- Příkazy musí končit středníkem
- Funkce musí být definovány až pod samotnými příkazy

- Pouze jednořádkové komentáře ( // komentář)
- Identifikátor nesmí obsahovat speciální znaky (kromě `_`) a začínat číslem

## 2.3 Konstrukce jazyka

### 2.3.1 Povinná

#### 2.3.1.1 Definice celočíselných proměnných

```
1 int a = 5;
```

#### 2.3.1.2 Definice celočíselných konstant

```
1 const int b = -8;
```

#### 2.3.1.3 Přiřazení

```
1 int a = 10;  
2 int b = a;
```

#### 2.3.1.4 Základní aritmetika a logika

Aritmetika:

```
1 int a = 3 ^ 3;  
2 int b = 50 % 12;  
3  
4 int c = a + b;  
5 int d = b - a;  
6 int e = b / a;  
7 int f = a * b;  
8 int g = ---a;  
9 int h = ++a;  
10 int i = (a + b - 10) * 5;
```

Operátory pro porovnání:

```
1 boolean a = 1 < 50;  
2 boolean b = 2 <= -5;  
3 boolean c = 5 >= 6;  
4 boolean d = 15 > 20;  
5 boolean e = 5 == 5;  
6 boolean f = 10 != 22;
```

### 2.3.1.5 Cyklus - while

```
1 int a = 0;
2 while (a < 10) {
3     a = a + 2;
4 }
```

### 2.3.1.6 Podmínka if (bez else)

```
1 int a = 18;
2 if (a != 0) {
3     a = a + 9;
4 }
```

### 2.3.1.7 Definice funkce a její volání

```
1 int res = foo();
2
3 func int foo() {
4     return 10;
5 }
```

## 2.3.2 Rozšiřující

### 2.3.2.1 Cyklus do-while

```
1 int a = 10;
2 do {
3     a = a + 8;
4 } while (a < 50);
```

### 2.3.2.2 Cyklus repeat-until

```
1 int a = 5;
2 repeat {
3     a = a + 1;
4 } until (a >= 10);
```

### 2.3.2.3 Cyklus for

```
1 for (int a = 0 to 10) {
2     ...
3 }
```



### 2.3.2.4 Else větev

```
1 if (false) {  
2     ...  
3 } else {  
4     ...  
5 }
```

### 2.3.2.5 Datový typ boolean a logické operace s ním

Aritmetika:

```
1 boolean a = true;  
2 boolean b = false;  
3  
4 boolean c = a && b;  
5 boolean d = a || b;  
6 boolean e = !a;  
7 boolean f = ((a || b) && b);
```

Operátory pro porovnání:

```
1 boolean a = true == false;  
2 boolean b = true != false;
```

### 2.3.2.6 Násobné přiřazení

```
1 int a = 5;  
2 int b = 6;  
3 int c = 10;  
4 int d = -10;  
5  
6 b = a = d = c = -80;
```

### 2.3.2.7 Parametry předávané hodnotou

```
1 int res = foo(3,4);  
2  
3 func int foo(int a, int b) {  
4     return a * b;  
5 }
```

### 2.3.2.8 Návrátová hodnota podprogramu

```
1 int num = 8;  
2 int res = foo(num);  
3  
4 func int foo(int a) {  
5     return a - 1;  
6 }
```

# Testování

3

## Závěr

4

## Seznam výpisů