

Překladač z jazyka daného specifikací do tříadresového kódu

Semestrální projekt z předmětu Programovací jazyky a překladače, LS 2016/17

Navrhněte, vytvořte, otestujte a zdokumentujte překladač programovacího jazyka X, jehož gramatika odpovídá následující slovní specifikaci:

- Program může být buď prázdný, nebo obsahuje neomezený počet nevnořených funkcí.
- Každá funkce musí obsahovat typ, identifikátor, argumenty a definici bloku.
- Typ je integer, ovšem gramatika musí být navržena tak, aby ji bylo možné rozšířit i o další typy.
- Identifikátor je programátorem zvolené jméno.
- Funkční argumenty jsou vždy uzavřeny normálními závorkami, tj. '(' a ')'. Uvnitř závorek je
 jejich seznam, který může být prázdný, případně obsahovat jeden nebo více prvků
 oddělených čárkou.
- Každý prvek seznamu argumentů obsahuje typ a svůj identifikátor, tj. programátorem zvolené jméno.
- Definice bloku je vždy uzavřena složenými závorkami, tj. '{' a '}'.
- Blok může být prázdný, případně obsahovat jeden nebo více řídicích příkazů.
- Řídicím příkazem může být:
 - o Definice bloku
 - o Podmíněný příkaz
 - o Příkaz while
 - Přiřazovací příkaz ukončený středníkem
 - o Návratový příkaz ukončený středníkem
- Podmíněný příkaz
 - o Obsahuje definici bloku nebo
 - Začíná klíčovým slovem IF a otvírací závorkou '(', následovanou podmínkou, které je ukončena zavírací závorkou ')', následovanou definicí bloku a alternativou podmínky
 - Alternativa podmínky může být buď prázdná, nebo začínat klíčovým slovem ELSE, následovaným definicí bloku.
- Příkaz while
 - Začíná klíčovým slovem WHILE a otvírací závorkou '(', následovanou podmínkou, které je ukončena zavírací závorkou ')', následovanou definicí bloku.
- Podmínka je ve tvaru:

podmínka relace výraz

nebo pouze

výraz

Relacemi jsou oba typy ostrých a neostrých nerovností (<, <=, >, >=) a absolutní rovnost a nerovnost (==, !=)

1/3 v.1.0, 28. 3.2017, JV



- Přiřazovací příkaz má strukturu typ identifikátor přiřazovací příkaz ('=') a výraz. V případě volitelné implementace více typů (např. integer a char) by měl obsahovat i typovou kontrolu
- Návratový příkaz je složen z klíčového slova RETURN, následovaného výrazem
- Výraz (v případě jediného typu integer pouze aritmetický) obsahuje identifikátor nebo číslo nebo výrazy se správně asociativně a prioritně uspořádanými operátory +, -, *, /.
 Součástí výrazu mohou být i obyčejné závorky, zavádějící vlastní prioritu jimi uzavřeného výrazu.

Překladač generuje mezikód v maximálním tvaru:

$$A = B \text{ op } C$$

kde A je cílová adresa, reprezentující buď trvalou (programátorem definovanou) nebo dočasnou (temporary) proměnnou Ti. B a C jsou volitelné operandy a *op* je operátor.

Pro realizaci řídicích příkazů je třeba generovat ještě kódové instrukce skoku ve tvaru:

IF podmínka GOTO návěští nebo GOTO návěští

včetně vytvoření prázdných řádků s odpovídajícím návěštím. Kromě toho musí být překladač schopen ignorovat neviditelné mezery, jednořádkové komentáře uvozené dvojitým lomítkem // a víceřádkové komentáře v klasické C notaci, tj. /* ... */.

Vzorový příklad *vzor*. *x* slouží k ilustraci jazyka i požadovaného výstupu. Zároveň bude použit jako jeden z testovacích příkladů pro odevzdané projekty.

Organizační poznámky

Výsledný překladač bude realizován pomocí standardních generátorů (např. typu flex a yacc) ve formě dvou souborů, lexikálního a syntaktického analyzátoru. Pomocné deklarace mohou být umístěny v souborech typu *.h.

Úplný a jednorázový překlad projektu zajistěte prostřednictvím utility *make*. Ke zdrojovým kódům dodejte nejméně 3 testovací příklady.

Překladač pracuje se standardním vstupem a výstupem (stdin, stdout).

Zajistěte, aby překladač po nalezení jakékoli syntaktické chyby nepřerušil svoji práci, ale zpracoval i zbytek vstupního kódu. Na konci pak vypíše co nevýstižnější informaci o chybách, které se v průběhu překladu vyskytly.

Kompletní projekty se budou odevzdávat v komprimované formě do e-Learning_ového kurzu xPJP. Archivní soubor bude obsahovat lexikální a syntaktický analyzátor (*.l, *.y), makefile, případné soubory *.h, tři testovací zdrojové kódy a projektovou dokumentaci, obsahující:

- a) Návrh gramatiky,
- b) Slovní charakteristiku dosažené funkčnosti,
- c) Ukázku vstupů a výstupů překladače (výpis, screenshot apod.).

2/3 v.1.0, 28. 3.2017, JV



Zdrojový kód a tříadresový výstup programu vzor.x:

```
Zdrojový kód
                                                  Výstupní tříadresový kód
// Komentar
                                                  T1 = a + b ;
                                                  c = T1;
/* I toto
                                                  T1 = a + b;
je komentar
                                                  T2 = c * d;
                                                  T1 = T1 - T2;
                                                  c = T1;
int prvni(){}
                                                  T1 = b - c;
                                                  T1 = T1 * d;
int druha(int a, int b){}
                                                  T1 = a + T1;
                                                  c = T1;
int treti(int a, int b){int c = a+b;}
                                                  T1 = 3 * 5;
                                                  T1 = T1;
int ctvrta(int a, int b){int c = a+b-c*d;}
                                                  T1 = T1 \leftarrow x;
                                                  if not T1 goto L1;
int pata(){int c = a+(b-c)*d;}
                                                  T1 = a;
                                                  T1 = T1 < 10;
int sesta(){
                                                  if not T1 goto L2;
if(3*5 <= x){
                                                  b = 20;
  if(a < 10){
                                                  goto L3;
   int b = 20;
                                                  L2:
                                                  b = 40;
 else{
                                                  L3:
    int b = 40;
                                                  goto L4;
                                                  L1:
                                                  b = x;
 }
                                                  T1 = a;
else{
                                                  T1 = T1 < 10;
 int b = x;
 while(a < 10){
  int a = 100;
                                                  if not T1 goto L6;
 }
                                                  a = 100;
 return b;
                                                  goto L5;
                                                  L6 :
}
                                                  L4:
```

Pro akceptování semestrálního projektu:

- a) Utilitou *make* musí být možno zkompilovat překladač do spustitelné podoby v prostředí cygwin,
- b) Překladač musí zpracovat příklad *vzor.x* přinejmenším na úrovni syntaktické analýzy.

Termín odevzdání: pátek, 26. 5. 2017, 12:00.

3/3 v.1.0, 28. 3.2017, JV