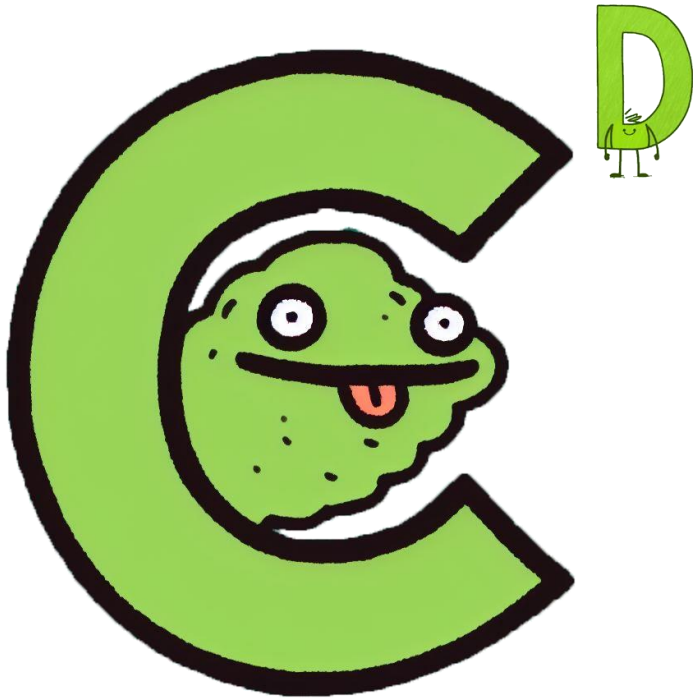


yadc

VLADIMÍRA KIMLOVÁ

DOMINIK ZAPPE

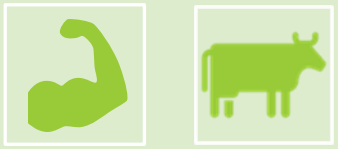
15. 1. 2024

Popis jazyka – základ

Silně staticky typovaný	
Datové typy:	int, bool, string, float, void, pointery, ~~ratio~~
Aritmetika:	+ - * / %
Přiřazení:	=
Porovnání:	== < > <= >= !=
Logické operátory:	&& || !
Řídící konstrukce:	while, for, if else, goto, řada dalších…

~~(Klíčová slova možná budou ve finální verzi změněna na atypická)~~

Technologie + Cílová platforma



flex + bison (lex + yacc)

Instrukce rozšířené PL/0

Zvolená bodovaná rozšíření – základní

Definice celočíselných proměnných a konstant
Přiřazení
Základní aritmetika a logika
Cyklus (while)
Jednoduchá podmínka
Definice podprogramu a jeho volání

Zvolená bodovaná rozšíření – 1 bod

Další typy cyklů (do while, until, repeat until, for)
Else větev
Datový typ boolean + operace s ním
Datový typ real + operace s ním
Datový typ string + operace s ním
Podmíněné přiřazení (ternární operátor)
Příkazy pro vstup a výstup (I/O operace)

Zvolená bodovaná rozšíření – 2 body

Příkaz goto
Pole a práce s jeho prvky
Operátor pro porovnání stringů
Parametry předávané hodnotou
Návratová hodnota z podprogramu
Anonymní vnitřní funkce

Zvolená bodovaná rozšíření – 3 body

Dynamicky přiřazovaná paměť a pointery

Parametry předávané odkazem

Vlastní rozšíření – ? bodů (1/2)

Chybové hlášky lexikálních, syntaktických a sémantických chyb
Vlastní AST – modulární implementace spojená s návrhovým vzorem visitor
Možnost deklarace globálních proměnných kdekoliv (klidně i mezi deklaracemi funkcí) (kdekoliv – myšleno v rámci globálního scopu)
Možnost deklarace lokálních proměnných kdekoliv (nejen na začátku bloku)
Možnost pouhé deklarace hlavičky funkce + dodeklarování těla později (inspirace v C)
Příkazy break a continue uvnitř cyklů
Návratová hodnota z hlavní funkce je zachycena v globálním bloku (bez globálních proměnných na adrese 3, jinak 3 + velikost datových typů globálních proměnných)

Vlastní rozšíření – ? bodů (2/2)

Vyřešení built-in funkcí, které se při použití ve zdrojovém kódu vygenerují na začátku instrukcí
Možnost explicitního přetypovávání
Implicitní přetypovávání intu na floaty, když se jedná o binární operaci, kde jeden z operandů je float (je možné implicitní přetypování přebít explicitním a rozbít si zásobník)
Jednoduchý type checking přiřazení, parametrů a return statementů
Optimalizace nad AST (zjednodušení aritmetiky a logiky pro binární operace)
Optimalizace nad vygenerovanými instrukcemi (zbavení se zbytečných skoků, které vedou na další nepodmíněné skoky)

Rozšíření webového interpreteru rozšířené instrukční sady PL/0 – ? bodů

Opravení I/O jako celku – instrukce WRI např. mazala znak ze vstupu
Opravení I/O při spouštění pomocí „run“ – I/O fungovalo pouze při krokování
Oprava nápovědy – PLD a PST byly chybně popsány (adresa a level jsou prohozeny)
Vylepšení o float – instrukce ITR, RTI a OPF (implementace instrukcí, vysvětlovač, nápověda)
Vylepšení o nenutnost číslování instrukcí (instrukce lze nyní psát bez explicitního očíslování – lépe se píše ručně pseudo assembly PL/0)
Vstupní pole nyní může přijímat speciální ASCII znaky jako CR, LF (použito jako ukončovací znak pro vstup)
Kompletní překlad nápovědy do češtiny

Testování

Python + Selenium WebDriver

Využití upravovaného interpreteru (přidali jsme ID k nějakým html elementům)

Založeno na I/O

Test case = instrukce, input, očekávaný output

Děkujeme za pozornost!