PARSAREA TIPURILOR DE DATE COMPUSE ALE LIMBAJULUI DE PROGRAMARE C

Cristian CEMÎRTAN

CZU: 004.6:004.43 cemirtan.cristian@outlook.com

In this thesis, an algorithm was developed to parse composite data types and their declarations in C program snippets. The declarations are then transformed into syntax tree nodes, which allow the compiler to decide which operations can be applied to them. For educational purposes, a visual program was also developed to illustrate how a syntax tree is seen from the point of view of a compiler.

Limbajul de programare C, fiind foarte actual și popular, poate fi utilizat în orice proces de dezvoltare a unui sistem informatic, de la compilatoare până la sisteme de operare [1]. Limbajul C suportă tipurile de date: fundamentale (numere și numere reale cu precizie finită) și compuse (structuri, masive cu dimensiune fixă, pointeri și funcții). Vorbind despre tipurile de date compuse, sintaxa lor este complexă deoarece în majoritatea cazurilor – ele nu pot fi citite de la stânga la dreapta, și că parantezele rotunde și pătrate sunt utilizate excesiv, creând confuzie multor programatori [2, p. 122].

În această lucrare, se propune de a elabora un automat finit ce parsează una sau mai multe declarații scrise în limbajul de programare C. Algoritmul automatului finit, ce recunoaște tipurile de date compuse în limbajul C, este implementat inițial în limbajul Prolog. Apoi este reimplementat în limbajul de programare C++ Common Language Interface, cunoscut ca C++/CLI, o extensie a limbajului C++ dezvoltat de Microsoft care permite integrarea tehnologiilor .NET în programele C++ [3, p. 12]. Din tehnologia .NET, am beneficiat de expresii regulate, obiecte gestionate și de clasele care sunt deseori întâlnite în limbajul de programare C#. Scopul utilizării tehnologiei .NET este sporirea eficienței programului software, eliminând complet necesitatea de a elibera explicit zonele de memorie care nu vor mai fi utilizate. Acest lucru înseamnă că gestionarea memoriei programului devine automatizată datorită colectorului de gunoi (Garbage Collector), un element esențial al tehnologiei .NET.

Pentru a implementa analiza lexicală, ca componentă pentru automatul finit, am beneficiat de utilizarea expresiilor regulate din tehnologia .NET. Conform documentațiilor de specialitate oferite de Microsoft, o expresie regulată reprezintă un șablon pe care motorul de căutare încearcă să-l potrivească cu textul introdus de la stânga la dreapta [4]. Motorul de căutare, sau motorul de expresii regulate, este reprezentat de clasa Regex din spațiul de nume System.Text.RegularExpressions [4].

Utilizând clasa Regex, automatul finit va analiza întregul text introdus de utilizator și apoi îl va împărți în bilete sau „corespondențe”, care sunt numite altfel instanțe

ale clasei Match din spațiul de nume menționat anterior, System.Text.RegularExpressions [4]. Șablonul expresiei regulate utilizate pentru parsare arată în felul următor:

```
(/\*((?'comment'/\*)|(?'-comment'\*/)|(?!\*/).)+\*/|(//|\#).*?(?<!\\\s*)$)|
\b((?'sign'(un)?signed)|
(?'length'short|long)|
(?'type'void|char|int|float|double)|
(?'sType'struct|union|enum)|
(?'qualifier'const|volatile|restrict)|
(?'specifier'static|auto|register|extern))\b|
(?'ellipsis'\.{3})|
(?'brace'[()])|
(?'arrayBrace'[[\]])|
(?'scopeBrace'[{}])|
(?'special'[,;*]|(typedef|inline)\b)|
(?'integer'\d+)|
(?'name'\w+)|
(?'unknown'""(\\""|.)*?""|\S)
```

Fig. 1. Șablonul expresiei regulate

Pentru a fonda baza experimentală a cercetării, pe partea front-end a programului în C++/CLI, a fost elaborat un alt program în limbajul de programare C# folosind framework-ul Windows Forms din tehnologia .NET. Programul vizual are un formular text unde va fi introdus un fragment de cod scris în limbajul C. După apăsarea butonului de confirmare (Parse), programul va genera un nou formular, conținând rezultatele procesului de parsare, cu două compartimente: forma textuală și forma arborescentă. În fig. 2, datele de intrare, utilizate pentru a testa algoritmul, au fost preluate din resursa proprie a autorului: https://github.com/cemti/cedacri-asdc/blob/master/Lab5/3.c.

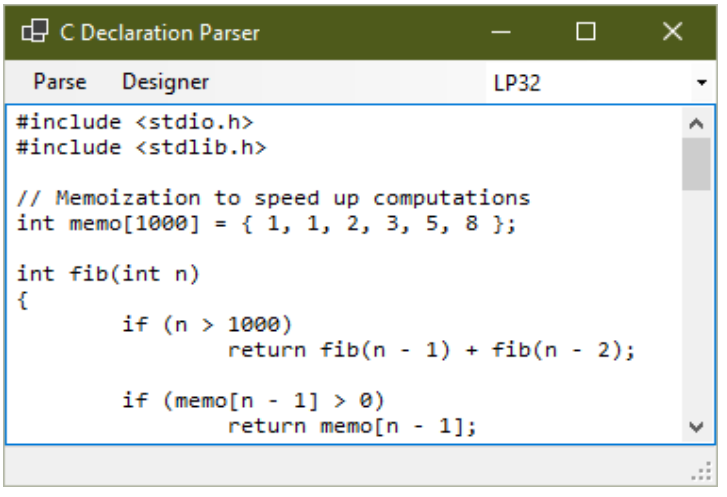


Fig. 2. Datele de intrare

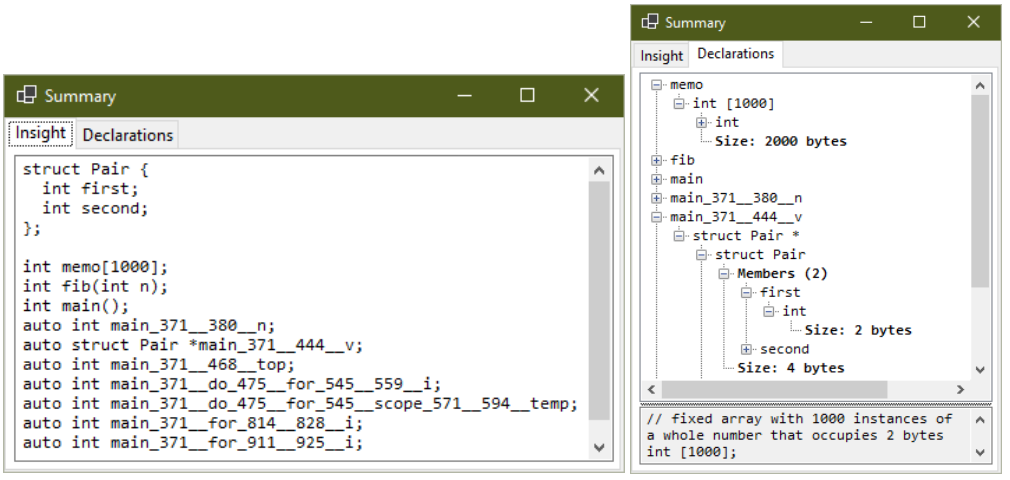


Fig. 3. Rezultatele programului

Aspectul principal al automatului finit, este de a împărți un tip de date compuse în mai multe „etaje” sau subtipuri, care vor fi procesate și inserate într-un arbore de sintaxă. Produsul software are oportunitatea de a fi folosit în mediile de dezvoltare destinate limbajelor de programare C și C++ (de exemplu, Visual Studio, CLion), unde programatorul va avea posibilitatea de a declara o variabilă fără să întâmpine dificultățile menționate în importanța temei elaborate.

Compilatoarele pot, de asemenea, să utilizeze algoritmul software-ului pentru a îmbunătăți procesul de compilare a programelor elaborate în limbajele menționate anterior. Produsul software, în întregime, este disponibil public pe GitHub: https://github.com/cemti/decl-parser, sub licența MIT.

Referințe:

1. CEMÎRTAN, C. *Algoritmul ce analizează sintactic tipurile de date compuse în limbajul C*. În: *Interuniversitaria*. Ediția 19, Vol.2, 4 mai 2023, Bălți, Republica Moldova: Universitatea de Stat „Alecu Russo” din Bălți, 2023, pp. 80-84. ISBN 978-9975-50-305-1.
2. KERNIGHAN, B. W., RITCHE, D. M. *The C Programming Language*, 2nd Edition, Pearson, 1988. 272 p. ISBN: 978-013-30-8621-8.
3. *Standard ECMA-372 1st Edition, C++/CLI Language Specification*, 2005. Disponibil: https://www.ecma-international.org/wp-content/uploads/ECMA-372_1st_edition_december_2005.pdf. Accesat la 14/03/23.
4. Microsoft Learn. *.NET regular expressions*, 2023. Disponibil: https://learn.microsoft.com/en-us/dotnet/standard/base-types/regular-expressions. Accesat la 11/05/24.

Recomandat
***Maria CAPCELEA**, dr., conf. univ.*