**AiFJ LV3**

Andreja Gojić

Andreja Nađ

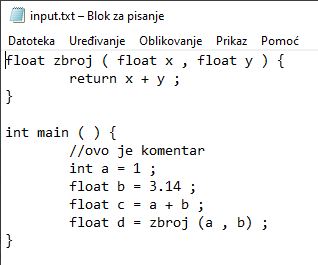
|  |  |
| --- | --- |
| **1. Osmisliti programski jezik koji se sastoji od različitih kategorija tokena, odnosno leksičkih klasa:** | |
| **a) identifikatori** | Ne sadrže ključne riječi. Sadrži slova a-Z i znamenke. |
| **b) ključne riječi** | “break”, “case”, “char”, “const”, “default”, “do”, “double”, “else”, “float”, “for”, “if”, “int”, “long”, “return”, “short”, “switch”, “unsigned”, “void”, “volatile”, ”while” |
| **c) separatori** | ; : , ‘ ‘ (=space) |
| **d) operatori** | a) aritmetički operatori: + - \* / %  b) relacijski operatori: < > <= >= == !=  c) logički operatori: ! && ||  d) operator pridruživanja: =  e) operator indeksa niza: [] |
| **e) konstante** | Počinju sa #define kojega slijede simboli ili nizovi brojeva, velikih i/ili malih slova. |
| **f) komentari** | Podržani su jednolinijski komentari koji počinju sa znakom // |

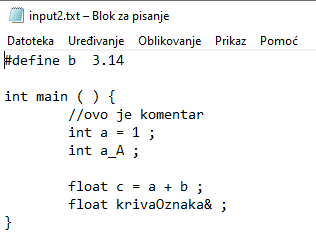
|  |  |  |
| --- | --- | --- |
| **Primjer** | | **Opis** |
| 1. float zbroj ( float x, float y ) {  2. return x + y;  3. } | | line 1: float zbroj ( float x, float y ) {  ('float', keywords)  (' ', separators)  (zbroj, identifiers)  (' ', separators)  ('(', unknown or value)  (' ', separators)  ('float', keywords)  (' ', separators)  ('x', identifiers)  (' ', separators)  (',', separators)  ('float', keywords)  (' ', separators)  ('y', identifiers)  (' ', separators)  (')', unknown or value)  (' ', separators)  line 2: return x + y;  ('', identifiers)  ('', operators)  (' ', separators)  ('return', keywords)  (' ', separators)  ('x', identifiers)  (' ', separators)  ('+', operators)  (' ', separators)  ('y', unknown or value)  (';', separators)  line 3: } |
| 1. int main ( ) {  2. int a = 1;  3. float b = 3.14;  4. float c = a + b;  5. float d = zbroj(a, b);  6. } | | line 1: int main ( ) {  ('int', keywords)  (' ', separators)  ('main', identifiers)  (' ', separators)  ('(', unknown or value)  (' ', separators)  (')', unknown or value)  (' ', separators)  line 2: int a = 1;  (' ', separators)  ('int', keywords)  (' ', separators)  ('a', identifiers)  (' ', separators)  ('=', operators)  (' ', separators)  ('1', unknown or value)  (';', separators)  line 3: float b = 3.14;  (' ', separators)  ('float', keywords)  (' ', separators)  ('b', identifiers)  (' ', separators)  ('=', operators)  (' ', separators)  ('3.14', unknown or value)  (';', separators)  line 4: int c = a + b;  (' ', separators)  ('int', keywords)  (' ', separators)  ('c', identifiers)  (' ', separators)  ('=', operators)  (' ', separators)  ('a', identifiers)  (' ', separators)  ('+', operators)  (' ', separators)  ('b', identifiers)  (';', separators)  line 5: int d = zbroj(a, b);  (' ', separators)  ('int', keywords)  (' ', separators)  ('d', identifiers)  (' ', separators)  ('=', operators)  (' ', separators)  (zbroj, identifiers)  ('(', unknown or value)  ('a', identifiers)  (',', separators)  ('b', identifiers)  (')', unknown or value)  (';', separators)  line 6: } |
| **2. Konstruirati odgovarajuće automate s konačnim brojem stanja za prepoznavanje pojedinih**  **kategorija.** | | |
| **a) identifikatori** |  | |
| **b) ključne riječi** |  | |
| **c) separatori** |  | |
| **d) operatori** |  | |
| **e) konstante** |  | |
| **f) komentari** |  | |
| **g) jezik** |  | |

**3. zadatak**

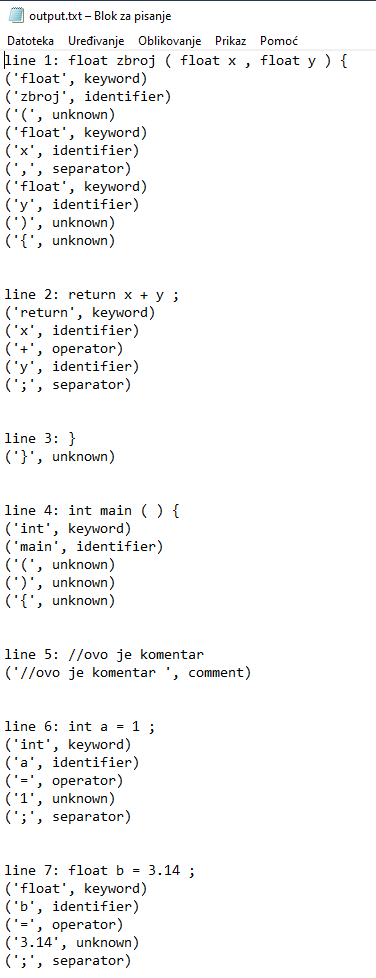
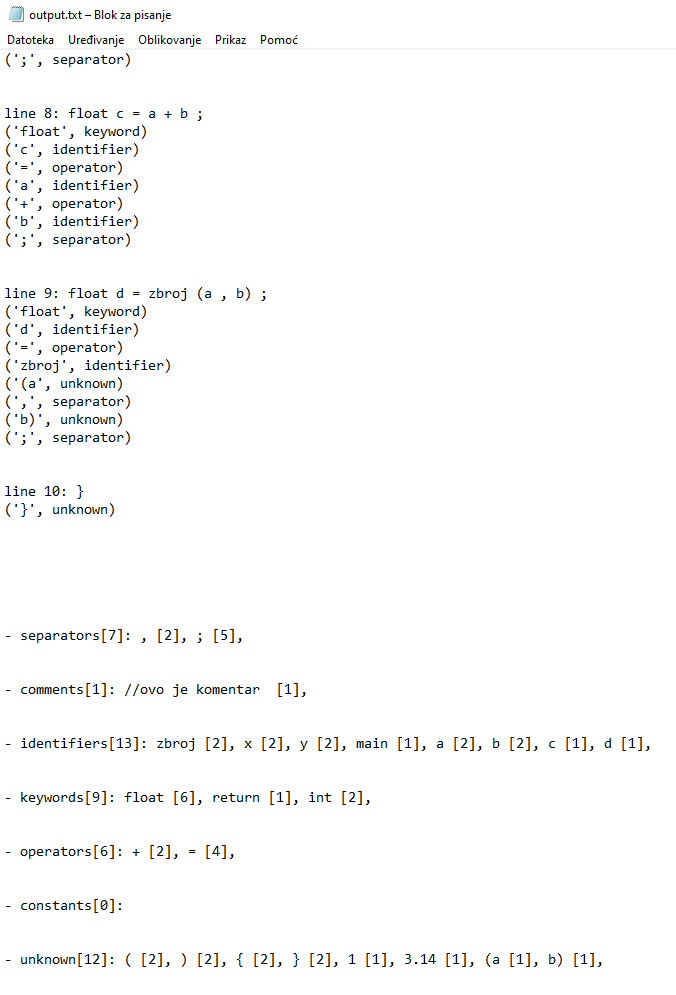
Kod: <https://github.com/Bileth/AiFJ_LV3>

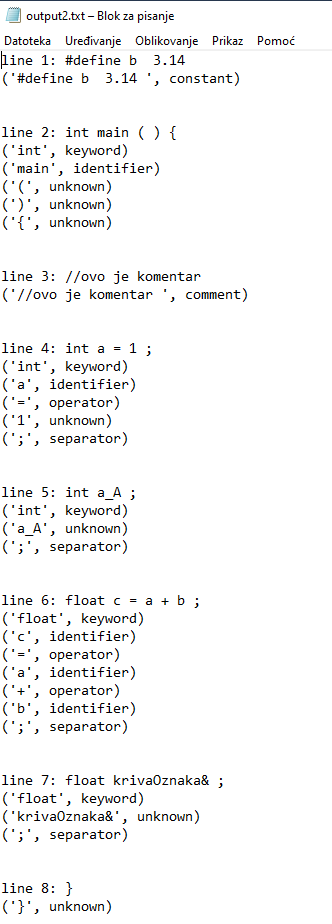
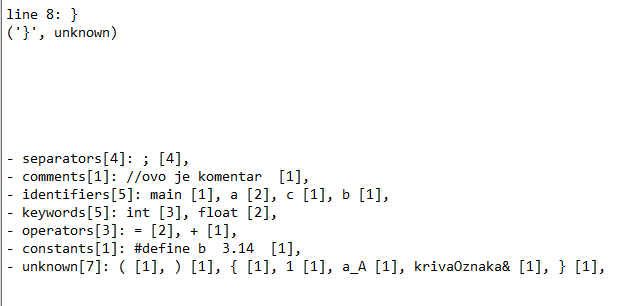
Ulazne datoteke:





Izlazne datoteke nakon obrade:

Unutar koda definirali smo vektore niza koji sadrže ključne riječi, operatore i separatore kako bi ih dalje mogli uspoređivati.

vector<string> myKeywords = { "break", "case", "char", "const", "default", "do", "double", "else", "float", "for", "if", "int", "long", "return", "short", "switch", "unsigned", "void", "volatile", "while" };

vector<string> myOperators = { "+", "-", "\*", "/", "%", "<", ">", "<=", ">=", "==", "!=", "!","&&", "||", "="};

vector<string> mySeparators = { ",", ":", ";", " "};

Vektor lines sadrži sve linije iz datoteke „input.txt“. Nakon čitanja iz nje, datoteku zatvaramo. Otvorili smo i „output.txt“ datoteku u koju će se kasnije zapisati naš ispis. Način ispisa je trunc što znači da će pri svakom otvaranju obrisati prethodno zapisano.

ifstream inputFile;

ofstream outputFile;

inputFile.open("input.txt");

outputFile.open("output.txt", ios::trunc);

vector<string> lines;

while (getline(inputFile, line)) {

if (line != "")

lines.push\_back(line);

}

inputFile.close();

Pošto sada imamo sve linije, potrebno je svaku liniju odvojiti po riječima i znakovima te se to kod nas odvija sa razmakom (delimiter). Unutar te podjele, imamo još jednu koja se bavi uklanjanjem tab praznine (delimiter2).

string delimiter = " ";

vector<vector<string>> splitLines(lines.size());

size\_t pos = 0;

string token;

string delimiter2 = " ";

size\_t pos2 = 0;

for (int i = 0; i < lines.size(); i++) {

string s = lines[i];

if (s.rfind(" ", 0) == 0) {

pos2 = s.find(delimiter2);

if (pos2 != std::string::npos)

{

// If found then erase it from string

s.erase(pos2, delimiter2.length());

}

}

while ((pos = s.find(delimiter)) != std::string::npos) {

token = s.substr(0, pos);

splitLines[i].push\_back(token);

s.erase(0, pos + delimiter.length());

}

splitLines[i].push\_back(s);

}

Svaki znak je spremljen u 2D vektorski niz tako da se u redu nalazi podijeljena prethodna linija. Time održavamo prethodni smisao teksta no sada imamo bolju podjelu za usporedbu.

for (int i = 0; i < splitLines.size(); i++)

{

string line;

string outputline;

Sljedeće dvije linije koda nam spajaju naš 2D vektorski red u jedan niz kako bi mogli ispisati svaki red.

for (std::vector<std::string>::const\_iterator ii = splitLines[i].begin(); ii != splitLines[i].end(); ++ii)

line += \*ii + " ";

outputline = "line " + to\_string(i + 1) + ": " + line;

if(outputline != " " || outputline != "\n")

outputText.push\_back(outputline);

for (int j = 0; j < splitLines[i].size(); j++)

{

Funkcija in\_array koja se nalazi ispod ovog dijela koda se bavi provjerom nalazi li se određeni niz u vektorskom nizu. Ako se nalazi, onda lako znamo prepoznati je li ključna riječ, operator ili separator.

if (in\_array(splitLines[i][j], myKeywords))

{

outputline = "('" + splitLines[i][j] +"', keyword)";

outputText.push\_back(outputline);

all\_keywords.push\_back(splitLines[i][j]);

}

else if (in\_array(splitLines[i][j], myOperators))

{

outputline = "('" + splitLines[i][j] + "', operator)";

outputText.push\_back(outputline);

all\_operators.push\_back(splitLines[i][j]);

}

else if (in\_array(splitLines[i][j], mySeparators))

{

outputline = "('" + splitLines[i][j] + "', separator)";

outputText.push\_back(outputline);

all\_separators.push\_back(splitLines[i][j]);

}

Drugi slučaj provjere čine komentari i konstante pošto imaju određene znakove s kojima moraju početi, prvo to testiramo. Ako se nalazi, ostatak reda je zanemaren od danje provjere i prisvojen je u komentar ili konstantu. Pošto zanemarujemo dio reda, moramo j varijablu povećati za veličinu tog reda bez početnog elementa pošto je varijabla j već na njemu.

else if (splitLines[i][j].rfind("//", 0) == 0)

{

string commentLine;

int skipCounter = 0;

for (std::vector<std::string>::const\_iterator ii = splitLines[i].begin(); ii != splitLines[i].end(); ++ii) {

commentLine += \*ii + " ";

skipCounter++;

}

outputline = "('" + commentLine + "', comment)";

outputText.push\_back(outputline);

all\_comments.push\_back(commentLine);

j = j + skipCounter-1;

}

else if (splitLines[i][j].rfind("#define", 0) == 0)

{

string constLine;

int skipCounter = 0;

for (std::vector<std::string>::const\_iterator ii = splitLines[i].begin(); ii != splitLines[i].end(); ++ii) {

constLine += \*ii + " ";

skipCounter++;

}

outputline = "('" + constLine + "', constant)";

outputText.push\_back(outputline);

all\_constants.push\_back(constLine);

j = j + skipCounter - 1;

}

Zadnja provjera su identifikatori i nepoznate stavke. U našem slučaju, identifikatori su nizovi svih slova i brojeva, to znači da identifikator ne smije biti samo broj, ako je onda se smatra unknown.

else if (!in\_array(splitLines[i][j], myOperators)) {

if (!in\_array(splitLines[i][j], myKeywords)) {

if (!in\_array(splitLines[i][j], mySeparators)) {

if (splitLines[i][j].find\_first\_not\_of("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890") != std::string::npos || isNumber(splitLines[i][j])) {

outputline = "('" + splitLines[i][j] + "', unknown)";

outputText.push\_back(outputline);

all\_unknown.push\_back(splitLines[i][j]);

}

else {

outputline = "('" + splitLines[i][j] + "', identifier)";

outputText.push\_back(outputline);

all\_identifiers.push\_back(splitLines[i][j]);

}

}

}

}

}

outputText.push\_back("\n");

}

Provjera je li niz samo broj se bavi putem funkcije isNumber koja provjerava je li svaki znak znamenka ili ne.

bool isNumber(const string& str)

{

for (char const& c : str) {

if (std::isdigit(c) == 0) return false;

}

return true;

}

Rad ranije spomenute funkcije in\_array prikazan je u nastavku, prolazi od početka do kraja vektorskog polja i gleda nalazi li se u njemu tražena vrijednost.

bool in\_array(const std::string& value, const std::vector<std::string>& array)

{

return std::find(array.begin(), array.end(), value) != array.end();

}

Osim ovih funkcija, još se koristi i findingAllWords funkcija koja za predani vektorski niz i predan tip podatka (ključna riječ, operator, komentar i sl.) traži i prebrojava koliko puta se koji element pronašao. Isto tako appenda se na string svaki od pronađenih riječi i koliko puta je pronađen, takav niz se vraća kako bi ga mogli ispisati.

string findingAllWords(vector<string> words, string type) {

string line;

vector<string> used(words.size());

int allCounter = 0;

for (int i = 0; i < words.size(); i++) {

if(!words[i].empty())

allCounter++;

}

line.append("- ");

line.append(type);

line.append("[");

line.append(to\_string(allCounter));

line.append("]: ");

for (int i = 0; i < words.size(); i++) {

int counter = 1;

string s = words[i];

if (!in\_array(s, used)) {

for (int j = i + 1; j < words.size(); j++) {

if (words[i] == words[j]) {

counter++;

}

}

line.append(s).append(" [").append(to\_string(counter)).append("], ");

}

used.push\_back(s);

}

return line;

}

Za konačni ispis svega, sve smo spremali u outputText vektorsku varijablu te smo na to nadodali posljednji potreban ispis putem prethodne funkcije.

outputText.push\_back("\n");

outputText.push\_back("\n");

outputText.push\_back(findingAllWords(all\_separators, "separators"));

outputText.push\_back("\n");

outputText.push\_back(findingAllWords(all\_comments, "comments"));

outputText.push\_back("\n");

Nakon toga, sve što je ostalo je ispisati tu varijablu u našu izlaznu datoteku.

if (outputFile.is\_open()) {

for (int i = 0; i < outputText.size(); i++)

{

if (!outputText[i].length() == 0) {

outputFile << outputText[i] << " ";

outputFile << endl;

}

}

outputFile.close();

}