Programski prevodioci: Vežbe 1 - rešenja zadataka

Sadržaj

1. Uvod

U ovom dokumentu su data rešenja zadataka koji su rađeni na prvim vežbama.

2. Rešenja zadataka

2.1. Zadatak 0

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

int main(void) {
    int ch;
    int state = 0;              // Promenljiva koje određuje stanje u kom se parser
                                // nalazi.
    char currentWord[15]="";    // Ograničena dužina reči na 15, zarad jednostavnosti.
    int capitalWord = 0;        // Da li je u pitanju reč sa velikim slovom.
    int i = -1;

    while(1) {
        switch(state) {
            // Čitanje karaktera.
            case 0: {
                ch = getc(stdin);
                if(ch == '.')
                {
                    // Konzumiranje flag-a.
                    state = 1;
```

```
            }
            else if(ch == ' ' || ch == '\n' || ch == '\t'){
                // Ako ništa još nije upisano u currentWord, znači da je niz
                // belina u pitanju, pa ignorišemo. Ako je i > -1, znači da
                // je unešena neka reč i potom belina, pa reč treba obraditi.
                if(i > -1){
                    if(capitalWord){
                        // Konzumiramo flag.
                        capitalWord = 0;
                        // Postavljamo stanje na CWORD.
                        state = 2;
                    }
                    else
                        // Postavljamo stanje na WORD.
                        state = 3;
                }
                else{
                    // Postavljamo stanje na čitanje upisa.
                    state = 0;
                }
            }
            else if(isupper(ch)){
                // Ukoliko nije pocetno veliko slovo i ukoliko je je vec zapoceta
                // rec, znaci da je zapoceta rec sa malim slovom, pa flag za
                // capitalWord treba da ostane nula.
                if(i > -1 && !capitalWord)
                    capitalWord = 0;
                else
                    capitalWord = 1;
                // Dodavanje procitanog karaktera u rec.
                currentWord[++i] = ch;
            }
            else if(islower(ch)){
                // Dodavanje procitanog karaktera u rec.
                currentWord[++i] = ch;
            }
            // Kad dođemo do kraja fajla, prekidamo program.
            else if(ch == EOF)
                return 0;

            else
                state = -1;
        }; break;

        case 1: {
            // Ako smo pročitali tačku, ukoliko je postojala reč pre nje,
            // tj. ako je i > -1, potrebno je tu reč ispisati.
            if(i > -1){
                if(capitalWord)
                    printf("\nCWORD\t%s\n", currentWord);
                else
```

```
                    printf("\nWORD\t%s\n", currentWord);
                // Resetujemo brojač za dužinu reči.
                i = -1;
                // Praznimo memorijske lokacije za trenutnu reč.
                memset(currentWord, '\0', sizeof currentWord);
            }
            // Ispisujemo i tačku posle reči, ukoliko je ona postojala.
            printf("\nDOT\t.\n");

            state = 0;
        }; break;

        case 2: {
            printf("\nCWORD\t%s\n", currentWord);
            // Resetujemo brojač za dužinu reči.
            i = -1;
            // Praznimo memorijske lokacije za trenutnu reč.
            memset(currentWord, '\0', sizeof currentWord);
            state = 0;
        }; break;

        case 3: {
            printf("\nWORD\t%s\n", currentWord);
            // Resetujemo brojač za dužinu reči.
            i = -1;
            // Praznimo memorijske lokacije za trenutnu reč.
            memset(currentWord, '\0', sizeof currentWord);
            state = 0;
        }; break;

        case -1: {
            printf("GRESKA: %c", ch);
            return -1;
        }; break;
    }
  }
}
```

2.2. Zadatak 1

```
0|[+-]?[1-9][0-9]*       { printf("broj: %s\n", yytext); }

0[xX][0-9a-fA-F]{1,4}    { printf("heksa broj: %s\n", yytext); }

[+-]?[0-9]+\.[0-9]*      { printf("realni broj: %s\n", yytext); }

[bB][rR][eE][aA][kK]     { printf("kljucna rec: %s\n", yytext); }
  /* ILI */
```

```
(?i:break)              { printf("kljucna rec: %s\n", yytext); }
```

2.3. Zadatak 2

```
"//".*
```

2.4. Zadatak 3

```
[+-]?[0-9]+F  { printf("%dC", ((atoi(yytext) - 32) * 5/9 )); }
```

Ukucati u terminalu `man atoi` i pročitati kako funkcija `atoi()` radi. Zašto nije neophodno izbrisati slovo `F` pre parsiranja broja?

2.5. Zadatak 4

```
"/*"([^*]|("*"+[^*/]))*"*"+"/" ①
```

Put do rešenja :

- Komentar je bilo koji karakter između /* i */

```
"/*".*"*/"
```

Problem: ne dozvoljava karakter za novi red

- Komentar je bilo koji karakter između /* i */ ili karakter za novi red

```
"/*"(.|"\n")*"*/"
```

Problem: greedy, kod između dva komentara?

- Komentar je bilo koji karakter osim zvezdice između /* i */

```
"/*"[^*]*"*/"
```

Problem: ne dozvoljava zvezdice unutar komentara

- Dozvolićemo zvezdice koje ne prati / jer je to oznaka za kraj komentara

```
"/*"([^*]|("*"+[^/]))*"*/"
```

Problem: greedy, više zvezdica na kraju ne smatra krajem komentara

- Dozvolićemo zvezdice koje ne prati / ni *

```
"/*"([^*]|("*"+[^*/]))*"*/"
```

Problem: strogo, više zvezdica na kraju komentara?