Universitatea „Petrol – Gaze” din Ploiești

Aplicație Flex și Bison – Convertor de unități de măsură

Realizat de: Coșereanu Emanuel, INFO an 2, grupa 40322

Coordonator: Lector Dr. Daniela Șchiopu

Cuprins

Introducere

Această aplicație a fost creată cu scopul de a ajuta utilizatorii să convertească mai simplu și mai rapid diferite unități de măsură, venind în întâmpinarea nevoilor tot mai frecvente din viața de zi cu zi. Într-o lume din ce în ce mai globalizată, în care comunicarea între țări și culturi diferite implică adesea utilizarea unor sisteme variate de măsură, devine esențială capacitatea de a efectua conversii corecte și eficiente.

Fie că este vorba despre înțelegerea unui text, interpretarea unor valori tehnice sau simpla comparație între standarde internaționale, aplicația **Convertor de unități de măsură** se dovedește un instrument util și indispensabil. Aceasta oferă utilizatorilor posibilitatea de a realiza conversii pentru o gamă variată de unități de măsură, incluzând: lungime, timp, energie, presiune, masă, volum, suprafață și temperatură.

Flex

Fișierul **convertor.l** este folosit pentru a recunoaște numere, unități de măsura și cuvinte cheie în cadrul aplicației de conversie de unități. Scannerul transformă textul de intrare într-o secvență de tokeni ce vor fi utilizați de analizatorul sintactic (**convertor.y**).

Declararea numerelor

Codul de mai jos prezintă cum a fost declarat tokenul **NUMAR**. Convertorul acceptă numere de tip float (numere reale).

```
[0-9]+(\.[0-9]+)? {
     yylval.fval = atof(yytext);
     return NUMAR;
}
```

Declararea unităților de măsură și a cuvântului de legatură

Codul de mai jos definește unitățile de măsură care au fost alese pentru lungime, timp, energie, presiune, masă, volum, suprafață și temperatură și cuvântul de legătura *IN*.

```
kilometri|mile|metri|picioare|centimetri|inchi|kilograme|pounds|gr
ame|uncii|secunde|minute|celsius|fahrenheit|kelvin|litri|galoane|m
etri_patrati|acri|calorii|kilocalorii|bari|atmosfere {
     yylval.sval = strdup(yytext);
     return UNITATE;
}

in {
     return IN;
}
```

Bison

În fișierul **convertor.y** s-a construit un parser ce gestionează conversii între unitățile de măsură. Mai jos sunt explicate câteva concepte tehnice.

Secțiunea de definiții Bison

- **Cod C inclus**

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
extern int yylex(void);
extern void yyerror(const char *s);
```

Sunt incluse biblioteci standard (`stdio.h`, `string.h`, `stdlib.h`) pentru funcționalități precum intrare/ieșire, manipulare de șiruri și alocare de memorie.

Funcțiile `yylex` (scannerul) și `yyerror` (gestionarea erorilor) sunt declarate extern.

- **Tipuri de date și uniune (`%union`)**

```
%union
{
    float fval;       // Valoare numerica de tip float
    char *sval;       // Sir de caractere
    struct
    {
        float value;  // Valoare calculata de tip float
        char *unit;   // Unitatea de masura asociata
    } result;
}
```

- `%union` definește o uniune utilizată pentru stocarea valorilor returnate de tokeni sau calculate.

- `result` este un struct utilizat pentru a reprezenta o valoare calculată și unitatea ei.

- **Definirea tokenilor și tipurilor de simboluri**

```
%type <result> program
%type <result> conversion
%token <fval> NUMAR
%token <sval> UNITATE IN
```

- Tokenii `NUMAR`, `UNITATE` și `IN` au asociați tipuri de date din uniunea definită.

- Non-terminalele `program` și `conversion` au tipul `result` pentru a transporta informații despre conversii.

Sectiunea de reguli de producție

Această secțiune definește gramatică și acțiuni asociate:

- **Programul principal**

```
program: program conversion '\n'
{
    printf("Rezultat: %.2f %s\n", $2.value, $2.unit);
}
| /* nimic */
{
    // Actiune goală (nu face nimic), ajuta sa pot scrie mai multe conversii fara sa inchid consola
}
;
```

- Regula `program` permite introducerea mai multor conversii pe rând.

- Conversia este procesată de regula `conversion`, iar rezultatul este afișat pe ecran.

- **Regula de conversie**

```
conversion: NUMAR UNITATE IN UNITATE
{
    if (strcmp($2, "kilometri") == 0 && strcmp($4, "mile") == 0)
    {
        $$.value = $1 * 0.621371;
        $$.unit = strdup($4); // Copiaza unitatea in care s-a facut conversia
    }
     // alte conversii....
```

```
    else
    {
        yyerror("Conversie necunoscuta!"); // mesaj pentru conversii
invalide
        $$.value = 0;
        $$.unit = strdup("eroare");
    }

};
```

- Regula procesează o conversie de forma `<NUMAR> <UNITATE> IN <UNITATE>`.

- Condițiile `if` verifică combinațiile valide de unități și efectuează conversia matematică corespunzătoare.

- În caz de eroare (unități neacceptate), se afișează un mesaj și se setează valori implicite pentru rezultat.

Secțiunea codului suplimentar

Această secțiune definește funcții precum **`yyerror`** și **`main`**.

- **Funcția `yyerror`**

```
void yyerror(const char *s)
{
    fprintf(stderr, "Eroare: %s\n", s);
}
```

- Afișează erori în cazul în care parserul întâmpină probleme (de exemplu, un input invalid).

- **Funcția principală main**

```
int main(void)
```

```
{
    printf("Conversii disponibile:\n");
    printf("kilometri <-> mile\n");
    printf("metri <-> picioare\n");
    printf("centimetri <-> inchi\n");
    printf("kilograme <-> pounds\n");
    printf("grame <-> uncii\n");
    printf("secunde <-> minute\n");
    printf("celsius <-> fahrenheit\n");
    printf("celsius <-> kelvin\n");
    printf("fahrenheit <-> kelvin\n");
    printf("litri <-> galoane\n");
    printf("metri_patrati <-> acri\n");
    printf("calorii <-> kilocalorii\n");
    printf("bari <-> atmosfere\n");

    printf("\nIntrodu conversia (ex: 10 kilometri in mile):\n");
    yyparse();  // Asteapta si proceseaza o linie de input
     return 0;
}
```

- Afișează utilizatorului informații despre conversiile disponibile.
- Așteaptă input prin apelul `yyparse`, care integrează scannerul și parserul.

Scopul și funcționalitatea parserului

Parserul gestionează conversii pentru unități de măsură, procesând expresii precum: **10 kilometri in mile**.

Tokenii sunt extrași de scanner și analizați de parser.

În funcție de unitățile recunoscute, se efectuează calcule matematice și se afișează rezultatul.

Concluzie

Aplicația „Convertor de unități de măsură” dezvoltată cu ajutorul Flex și Bison reprezintă o soluție eficientă pentru gestionarea conversiilor între diferite unități de măsură. Implementarea sa simplă și concisă permite integrarea de noi unități și conversii, ceea ce ii conferă flexibilitate și utilitate într-o gamă largă de scenarii.