Eksperyment programistyczny, zadania.
Przedmiot: Języki Formalne i Kompilatory

Eliminacje 1
Zadanie do wykonania w technologii LEX / YACC.

Cel zadania:
Celem zadania jest napisanie analizatora maksymalnej zajętości pamięci operacyjnej programu w podzbiorze języka C. Analizator ma za zadanie obliczyć rozmiar danych globalnych oraz maksymalną zajętość stosu w podprogramach.
Należy założyć, że na wejściu znajduje się poprawny składniowo program, a zachowanie analizatora dla błędnego wejścia jest nieokreślone.
W pliku wejściowym znajdują się ciągi deklaracji zmiennych na przemian ze szkieletami funkcji.
W rozwiązaniach NIE wolno korzystać ze zmiennych globalnych.

Przyrost 1.
Skonstruować analizator dla ciągów deklaracji pojedynczych zmiennych. W pliku wejściowym znajduje się co najmniej jedna deklaracja.

Dopuszczalne są następujące typy :
- *char* – rozmiar 1 bajt,
- *int* – rozmiar 4 bajty,
- *float* – rozmiar 6 bajtów,
- *double* – rozmiar 8 bajtów.

Pomiędzy jednostkami leksykalnymi mogą znajdować się dowolnej długości ciągi spacji, tabulatorów i znaków nowej linii, które nie mają wpływu na znaczenie programu.
Każda deklaracja zbudowana jest w następujący sposób :
- słowo kluczowe (nazwa typu) *char*, *int*, *float*, *double*,
- identyfikator,
- znak ';'.

w odpowiedzi powinniśmy uzyskać tekst o postaci :
`Global data size:` *`liczba`*
gdzie *liczba* jest sumą rozmiarów poszczególnych zmiennych zadeklarowanych w programie.
Podstawowe wzorce :
identyfikator : *[_a-zA-Z][_a-zA-Z0-9]**

Przykłady :
wejście:
```
int _a;
```
wyjście:
```
Global data size: 4
```
wejście:
```
double _b0;
```
wyjście:
```
Global data size: 8
```
wejście:
```
int a;    float b;    float c; char __a;
double _g_;
```
wyjście:
```
Global data size: 25
```
wejście:
```
double _b0
```
wyjście:
```
Syntax error in line 1
```

Przyrost 2.
Rozbudować analizator o możliwość przetwarzania deklaracji list zmiennych. W pliku wejściowym znajduje się co najmniej jedna deklaracja.
Każda deklaracja zbudowana jest w następujący sposób :

- słowo kluczowe (nazwa typu) *char*, *int*, *float*, *double*,
- co najmniej jeden identyfikator (jeżeli jest ich więcej są one rozdzielone znakiem ','),
- znak ';'.

w odpowiedzi powinniśmy uzyskać tekst o postaci :
`Global data size: liczba`
gdzie *liczba* jest sumą rozmiarów poszczególnych zmiennych zadeklarowanych w programie.

Przykłady :

wejście:
```
int a,a,a;
```

wyjście:
```
Global data size: 12
```

wejście:
```
int a,b,c;
char a,b;
```

wyjście:
```
Global data size: 14
```

wejście:
```
int a, b;
float c, d e;
```

wyjście:
```
Syntax error in line 2
```

Przyrost 3.

Rozbudować analizator o możliwość przetwarzania wskaźników. Należy założyć, że wskaźnik to zmienną poprzedzona dowolną liczbą gwiazdek. Wskaźnik na zmienną dowolnego typu ma rozmiar 4 bajtów.

Przykłady :

wejście:

```
int a,*b;
```

wyjście:

```
Global data size: 8
```

wejście:

```
char a,*b,c;
```

wyjście:

```
Global data size: 6
```

wejście:

```
char a,*b,***c;
```

wyjście:

```
Global data size: 9
```

Przyrost 4.
Rozbudować analizator o możliwość przetwarzania tablic jedno- i wielowymiarowych.

Przykłady :

wejście:

```
char *a[1][2],c;
```

wyjście:

```
Global data size: 9
```

wejście:

```
float a,*a,a[1],*a[1];
```

wyjście:

```
Global data size: 20
```

wejście:

```
double *****z[2];
```

wyjście:

```
Global data size: 8
```

Przyrost 5.
Rozbudować analizator o możliwość przetwarzania zmiennych typu *struct* (bez etykiety typu).

Przykłady :

wejście:

```
struct
{
  int a;
};
```

wyjście:

```
Global data size: 0
```

wejście:

```
struct
{
  int a;
} a1;
```

wyjście:

```
Global data size: 4
```

wejście:

```
struct
{
  char a;
} *a1;
```

wyjście:

```
Global data size: 4
```

wejście:

```
struct
{
  char a;
  int c;
  struct
  {
    float f;
  }k;
} *a1[40], c;
```

wyjście:

```
Global data size: 171
```

Przyrost 6.
Rozbudować analizator o możliwość przetwarzania zmiennych typu *union* (bez etykiety typu).

Przykłady :

wejście:

```
union
{
  char a;
  int c;
};
```

wyjście:

```
Global data size: 0
```

wejście:

```
union
{
  char a;
  int c;
} a1;
```

wyjście:

```
Global data size: 4
```

wejście:

```
struct
{
   int c;
   union
   {
     char a;
     int ca;
   } a;
}j;
```

wyjście:

```
Global data size: 8
```

Przyrost 7.
Rozbudować analizator o możliwość analizy szkieletu definicji pojedynczej funkcji (funkcja zawiera tylko deklaracje zmiennych). Należy założyć, że każda funkcja ma co najmniej jeden parametr.
W odpowiedzi należy podać, oprócz rozmiaru danych globalnych, rozmiar stosu wymagany przez funkcję.

Przykłady :

wejście:
```
int funa(char a)
{
};
```

wyjście:
```
funa stack size: 1
Global data size: 0
```

wejście:
```
int funa(char *a, int a)
{
};
int c;
```

wyjście:
```
funa stack size: 8
Global data size: 4
```

wejście:
```
char *h;
int funb(char *a, int a)
{
  int a, *b;
  char c;
};
int a[1];
```

wyjście:
```
funb stack size: 17
Global data size: 8
```