Wstęp do programowania w języku C

Marek Piotrów - Wykład 5 - Instrukcje i gramatyka języka C

9 listopada 2016

Rodzaje instrukcji w języku C

- Instrukcja wyrażeniowa: *wyrażenie;*
- Instrukcja złożona (blok): **{ ... }** .
- Instrukcje warunkowe: **if** oraz **if-else**.
- Instrukcja selekcji: **switch**.
- Instrukcje powtarzania (pętli): **while**, **do-while** oraz **for**.
- Instrukcje skoku: **break**, **continue**, **return** oraz **goto**.

Instrukcje mogą być poprzedzane etykietami z dwukropkiem na końcu. Etykietami są **case** i **default** - wewnątrz instrukcji selekcji oraz identyfikatory - dla oznaczenia miejsca docelowego **goto**.

Prosty kalkulator

ZADANIE: Napisać program, który czyta ze standardowego wejścia ciąg liczb całkowitych dziesiętnych oddzielonych operatorami arytmetycznymi i drukuje wyliczoną wartość wyrażenia.

- Rozważamy tylko cztery podstawowe operatory: dodawania (+), odejmowania (-), mnożenia (*) i dzielenia całkowitego (/).
- Dla uproszczenia zakładamy, że mają one jednakowy priorytet i są lewostronnie łączne. Liczby mogą być poprzedzone znakiem plus (+) lub minus (-), który musi przylegać do liczby.
- Dla wygody wprowadzającego wyrażenie przyjmujemy, że elementy wyrażenia mogą być od siebie oddzielone dowolną liczbą tzw. białych znaków (spacji, tabulacji, znaków nowego wiersza, itd.). Na końcu wprowadzony musi być znak równości (=).

Prosty kalkulator - prostykalk.c I

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

/***************  OBLICZ: bardzo prosty kalkulator  *********************
 * Program czyta z wejscia ciag liczb calkowitych oddzielonych operacjami  *
 * arytmetycznymi: +, -, *, / i wykonuje je kolejno od lewej do prawej.    *
 * Koncem danych jest znak: =.                                             *
 ***************************************************************************/

/***************** PROTOTYPY FUNKCJI ******************/

void blad(const char komunikat[]);
static long long int czytaj_arg(void);
static int czytaj_op(void);
static long long int wyrazenie(void);

/***************** DEFINICJE FUNKCJI ******************/

void blad(const char komunikat[])
{
   printf("\nBŁĄD: %s\n\n",komunikat);
   exit(1);
}
```

Prosty kalkulator - prostykalk.c II

```c
int main(void)
{
   long long int wartosc;

   wartosc=wyrazenie();

   int ost_znak=czytaj_op();
   if (ost_znak != '=' && ost_znak != EOF)
     blad("Zły znak w wyrażeniu lub brak znaku = na jego końcu");

   printf("%lld\n",wartosc);
   return 0;
}

static int czytaj_op(void)
{
   int znak;

   do
      znak=getchar();
   while (znak != EOF && isspace(znak)) ;
   return znak;
}
```

Prosty kalkulator - prostykalk.c III

```
static long long int czytaj_arg(void)
{
   long long int liczba=0;
   int znak, minus;

   do
      znak=getchar();
   while (znak != EOF && isspace(znak)) ;

   minus=(znak == '-' ? -1 : 1);
   if (znak == '+' || znak == '-')
      znak = getchar();

   if (isdigit(znak)) {
      do {
         liczba = 10 * liczba + (znak-'0');
         znak = getchar();
      } while (isdigit(znak));
      ungetc(znak,stdin);
   }
   else blad("brak argumentu");
   liczba *= minus;
   return liczba;
}
```

Prosty kalkulator - prostykalk.c IV

```
static long long int wyrazenie(void)
{
   long long int arg1,arg2;
   int op;

   arg1=czytaj_arg();
   while ((op=czytaj_op()) == '+' || op == '-' || op == '*' || op == '/') {
      arg2=czytaj_arg();
      switch (op) {
         case '+':
            arg1+=arg2;
            break;
         case '-':
            arg1-=arg2;
            break;
         case '*':
            arg1*=arg2;
            break;
         case '/':
            arg1/=arg2;
            break;
      }
   }
   ungetc(op,stdin);
   return arg1;
}
```

Przykład użycia instrukcji: continue

```
    .
    .
    .

  for (i=0; i < n; ++i) {
      if (a[i] < 0)
          continue;   /* pomin element ujemny */

    /* przetwarzaj element nieujemny */

  ...

}


    .
    .
    .
```

Przykład użycia instrukcji: goto

```
    .
    .
    .

for (i=0; i < n; ++i)
    for (j=0; j < m; ++j)
        if (a[i] == b[j])
            goto znaleziono;

/*  nie znaleziono pary identycznych znakow */
...

znaleziono:

/* jest para identycznych znakow a[i]=b[j] */


    .
    .
    .
```

Gramatyka instrukcji języka C

instrukcja :
instrukcja-etykietowana
instrukcja-wyrażeniowa
instrukcja-złożona
instrukcja-wyboru
instrukcja-powtarzania
instrukcja-skoku
instrukcja-etykietowana :
identyfikator **:** *instrukcja*
case *wyrażenie-stałe* **:** *instrukcja*
default : *instrukcja*
instrukcja-wyrażeniowa :
$wyrażenie_{opc}$ **;**

Gramatyka instrukcji języka C - część druga

instrukcja-złożona :
    { *lista-elementów-bloku*$_{opc}$ **}**
lista-elementów-bloku :
    element-bloku
    lista-elementów-bloku *element-bloku*
element-bloku :
    deklaracja
    instrukcja
instrukcja-wyboru :
    if (*wyrażenie***)** *instrukcja*
    if (*wyrażenie***)** *instrukcja* **else** *instrukcja*
    switch (*wyrażenie***)** *instrukcja*

Gramatyka instrukcji języka C -część trzecia

instrukcja-powtarzania :

while (*wyrażenie***)** *instrukcja*

do *instrukcja* **while (** *wyrażenie***) ;**

for ($wyrażenie_{opc}$ **;** $wyrażenie_{opc}$ **;** $wyrażenie_{opc}$**)**

instrukcja

for (*deklaracja* $wyrażenie_{opc}$ **;** $wyrażenie_{opc}$**)** *instrukcja*

instrukcja-skoku :

goto *identyfikator* **;**

continue ;

break ;

return $wyrażenie_{opc}$ **;**

Definicja gramatyki bezkontekstowej

Jeśli A jest skończonym zbiorem liter (alfabetem), to przez A^* będziemy oznaczać zbiór wszystkich słów (skończonych ciągów liter) nad alfabetem A.

Gramatyka bezkontekstowa to czwórka (Σ, V, P, S), gdzie

- Σ jest skończonym alfabetem terminalnym (podstawowym).
- V jest skończonym alfabetem nieterminalnym (pomocniczym).
- $P \subseteq V \times (\Sigma \cup V)^*$ jest skończonym zbiorem produkcji.
- $S \in V$ jest symbolem początkowym (startowym).

Produkcję $(A, \beta) \in P$ zapisujemy $A \to \beta$.

Język generowany przez gramatykę bezkontekstową

Niech $G = (\Sigma, V, P, S)$ będzie gramatyką bezkontekstową.

Wyprowadzeniem w gramatyce G nazywamy relację:

$\Rightarrow_G \subseteq (\Sigma \cup V)^* V (\Sigma \cup V)^* \times (\Sigma \cup V)^*$ taką, że $w \Rightarrow_G u$ wtedy i tylko wtedy, gdy istnieją $A \in V$ oraz $\alpha, \beta, \gamma \in (\Sigma \cup V)^*$ spełniające następujące warunki:

- $(A \to \beta) \in P$;
- $w = \alpha A \gamma$;
- $u = \alpha \beta \gamma$.

Przez $\Rightarrow_G^*$ będziemy oznaczać tranzytywne i zwrotne domknięcie relacji $\Rightarrow_G$.

Gramatyka G definiuje język $L(G) = \{w \in \Sigma^* : S \Rightarrow_G^* w\}$.

Przykłady języków i gramatyk bezkontekstowych

Przykład 1

Niech $G_1 = (\{a, b\}, \{S\}, \{(S \to a\ S\ b), (S \to \epsilon)\}, S)$. Wtedy $L(G) = \{\epsilon, ab, aabb, aaabbb, \ldots\} = \{a^n b^n : n \geq 0\}$

- Przykład wyprowadzenia: $S \Rightarrow_{G_1} a\ S\ b \Rightarrow_{G_1} a\ b$.
- Przykład wyprowadzenia: $S \Rightarrow_{G_1} \epsilon$.

Jeśli gramatyka jest ustalona, to możemy opuszczać indeks gramatyki w wyprowadzeniu, np. $S \Rightarrow a\ S\ b \Rightarrow a\ b$.

Przykład 2

Niech $G_2 = (\{[,]\}, \{S\}, \{(S \to [S]S), (S \to \epsilon)\}, S)$. Wtedy $L(G)$ jest językiem prawidłowo rozstawionych nawiasów kwadratowych.