**Projekt języka mklab**

**1.Opis systemu**

Celem zadania jest opracowanie kompilatora mklab języka c podobnego umożliwiającego realizacje prostych zadań.

**2.Opis Funkcjonalności**

2.1 Obsługiwane typy danych:

„int” - zmienne całkowite

„dou” - zmienne wymierne, Wartość całkowita musi być oddzielona od wartości rzeczywistej znakiem „.”.

„typ\_zmiennej[n]” - zmienne tablicowe jednowymiarowe, alokowane dynamicznie

„typ\_zmiennej[ln][ln]...” - zmienne tablicowe maksymalnie 3 wymiarowe, alokowane statycznie

„str zmienna „string”” – zmienna tekstowa do której przypisany jest tekst „string”

Deklaracja zmiennej musi nastąpić po zapisie typu, nazwa nie może zawierać spacji, oraz zaczynać się od cyfry, oraz zawierać String, Stack (zarezerwowane dla kompilatora), może składać sie ze liter alfabetu, znaku "\_" oraz cyfr (ale nie może się od nich zaczynać).

Typy danych nie są integralne ze sobą. W razie konwersji możliwa jest utrata dokładności.

Przypisanie wartości może nastąpić dopiero w kolejnej lini.

Domyślnie zmienna ma wartość 0.

Zmienna moze zostac zadeklarowana pare razy w kodzie. Typ ostatniej deklaracji zostaje zapisany.

\*ln – liczby naturalne większe niż 0

2.2 Wyrażenia arytmetyczne:

„=” – przypisanie wartości

„+” – dodawanie wartości

„-” – odejmowanie wartości

„\*” – mnożenie wartości

„/” – dzielenie wartości

() – nawiasy wyznaczające kolejność działań

2.2 Wyrażenia logiczne:

„|” – or

„&” – and

„!” – negacja

„?” – czy rowne (==)? 1 kiedy równe, w innym przypadku 0

„>” - 1 kiedy lewa strona jest większa, w innym przypadku 0

„<” - 1 kiedy prawa strona jest większa, w innym przypadku 0

2.3 Spełnienie warunku:

Warunek jest spełniony jeśli wynik wyrażenia nie jest równy 0.

2.4 Wyrażenia warunkowe:

„if (warunek); {

…

};”

Wykonanie kodu, tylko jeśli warunek jest spełniony, warunek może składać się z dowolnej ilości, dowolnych wyrażeń arytmetycznych i/lub logicznych.

„if…

else; {

…

};”

Wykonanie kodu, jeśli warunek jest nie spełniony.

Konstrukcje 2.4 mogą być dowolnie zagnieżdżone

2.5 Pętlę:

„while; (warunek);

{

…

}”

w przypadku braku warunku wykonanie ciala funkcji zalezy od ostatniej operacji artmetycznej lub logicznej - jesli bedzie ona rowna 0 petla sie przerwie

Kod zapisany w ciele pętli wykonuję się, do kiedy warunek jest prawdziwy. Sprawdzenie warunku następuje przed wykonaniem kodu z ciała pętli.

Warunek może składać się z dowolnej ilości, dowolnych wyrażeń arytmetycznych i/lub logicznych.

Konstrukcje 2.5 mogą być dowolnie zagnieżdżone

2.6 Tablice

zmienna[index][index]… – poruszanie się po tablicy

2.7 Rozmiar tablicy

rozmiar tablicy odczytujemy #zmienna

2.8 Funkcje

fun nazwa\_funkcji

{

…

}

Funkcja nie może przyjmować ani zwracać parametrów.

Wywolanie funkcji:

fun nazwa\_funkcji;

* 1. Wczytywanie i wypisywanie wartości

„>> „tekst” – wypisanie znaków tekst]

„>> zmienna"” – wypisanie wartości zmiennej

">>"zmienna a="a"\n"" - zapis dozwolony

„a <<” - wczytanie wartości zmiennej (zmienna musi być wcześniej zadeklarowana)

2.10 Struktury

Deklaracja struktury:

struct nazwa\_struktury { typ zmienna; ... }

Stworzenie zmiennej struktury:

struct nazwa\_struktury nazwa\_zmiennej;

3. Gramatyka języka

Polecenia muszą kończyć się znakiem „;”.

Kompilator ignoruje znaki whitespace.