Учебный язык программирования MiniC для построения транслятора для дисциплины «Языки программирования и методы трансляции», «Практикум по программированию»

Учебный язык является подмножеством C++ с добавлением двух новых конструкций для ввода и вывода данных.

Язык поддерживает два знаковых типа данных: однобайтный char, однобайтный int. Отличие двух типов данных проявляется при выводе на печать: значение переменной типа char выводится как символ, а int – как число. Еще одна разница: char – беззнаковый тип с диапазоном [0…255], int – знаковый тип с диапазоном [-128…127].

Язык поддерживает одномерные массивы.

Язык поддерживает следующие константы:

- целочисленные: 3, -10, 0
- строковые: “Hello, World!”
- символьные: ‘H’, ‘i’, ‘\n’

Язык поддерживает неявное преобразование типов:

- при присваивании значение преобразуется к типу переменной-приемника
- в арифметических выражениях все значения преобразуются к типу int

Язык поддерживает локальные (стековые) и глобальные переменные.

Язык поддерживает только две области видимости – глобальную и локальную для функции. Все переменные, объявленные внутри функции, независимо от места объявления, видны везде в этой функции.

Язык поддерживает выражения с использованием скобок и всех перечисленных ниже операций с учетом их приоритетов. Все операции левоассоциативны. Цепочное присваивание типа a = b = c не поддерживается. Оператор присваивания не являться частью выражения.

Язык поддерживает скобочную структуру любого уровня вложенности и следующие операции.

Арифметические и присваивание:

Оператор	Синтаксис	Приоритет
Присваивание	a = b	
Сложение	a + b	4
Вычитание	a – b	4
Умножение	a * b	3
Деление	a / b	3
Унарный минус	-a	2

Операция взятия остатка от деления	a % b	3
Префиксный инкремент	++a	1
Постфиксный инкремент	a++	1
Префиксный декремент	--a	1
Постфиксный декремент	a--	1

Операции сравнения:

Оператор	Синтаксис	Приоритет
Равенство	a == b	6
Неравенство	a != b	6
Больше	a > b	5
Меньше	a < b	5
Больше или равно	a >= b	5
Меньше или равно	a <= b	5

Логические операции:

Оператор	Синтаксис	Приоритет
Логическое отрицание	!a	2
Логическое И	a && b	7
Логическое ИЛИ	a || b	8

Операции работы с массивами:

Оператор	Синтаксис	Приоритет
Обращение к элементу массива	a[b]	1

Логические выражения используют следующие соглашения: ноль является ложью, любое ненулевое значение – истиной.

Оператор объявления переменных

```
int a;
char a, b=5, c;
int mas[15];
char buf[] = “Some string”;
const about_pi = 3;
```

Комментарии:

```
/*
 это комментарий, который может состоять
 из нескольких строчек
 */
// вся оставшаяся часть строки является комментарием
```

Пустой оператор

```
;
```

Пустой оператор не совершает никаких действий и может находиться в любом месте программы.

Последовательности вычислений и блоки

Несколько идущих подряд операторов образуют *последовательность операторов*. Инструкции, если иное не указано в описании, заканчиваются точкой с запятой. Операторы могут быть сгруппированы в специальные вычислительные блоки следующего вида:

```
{
  (последовательность инструкций)
}
```

ограниченные при помощи двух разделителей:

- левая фигурная скобка ({) обозначает начало вычислительного блока,
- правая фигурная скобка (}) обозначает конец вычислительного блока.

Вычислительный блок называют ещё *составным оператором*. Всюду далее под названием оператор подразумевается простой или составной оператор. Перечисление операторов через запятую не поддерживается.

Блоки кода могут быть вложены друг в друга, но это не влечет за собой создание областей видимости переменных.

Условные операторы

```
if((условие)) (оператор)

if((условие)) (оператор)
else (альтернативный оператор)

switch((выражение)) {
case значение1: (оператор)
…
case значениеN: (оператор)
default: (оператор)
}
```

Операторы выполнения цикла

```
while(условие) [тело цикла]

do [тело цикла] while( условие)

for( блок инициализации;условие;оператор) [тело цикла]
```

блок инициализации содержит либо одно присваивание, либо ничего (объявления недопустимы)

блок оператора содержит либо один оператор инкремента, либо одно присваивание, либо ничего

Другие операторы перехода связаны с циклами и позволяют прервать выполнения тела цикла:

• оператор **break** немедленно прерывает выполнение тела цикла, и происходит передача управления на оператор, следующий непосредственно сразу за оператором цикла;
• оператор **continue** прерывает выполнение тела цикла и передаёт управление в начало цикла, что инициирует проверку условия цикла.

Оператор возврата из функции

```
return [значение];
```

Функция не может иметь тип void, то есть всегда должна возвращать какое-либо значение.

Объявление функции

```
[описатель] [имя] ( [список] );,
```

Определение функции

```
[описатель] [имя] ( [список] ) [тело]
```

Вызов функции

```
[имя] ( [список] );
```

Вызов функции может быть как сам по себе, так и являться частью выражения.

Искусственные операторы ввода/вывода

Ввод значения с клавиатуры в переменную

```
in a;
```

Вывод значения из переменной

```
out a;
```

Печать текста на экран. Оператор печати не переводит строку на начало.

```
out “Hello, world!”;
```

Таблица лексем языка MiniC

№	Лексемы	Кодовое имя	Описание
Лексемы со значением			
1	121, -1, …	num	Числовая константа
2	‘a’, …	chr	Символьная константа
3	“Hello, world!”, …	str	Строковая константа
4	i, j, func, …	id	Идентификатор
Скобки и пунктуация			
5	(	lpar	Открывающая круглая скобка
6	)	rpar	Закрывающая круглая скобка
7	{	lbrace	Открывающая фигурная скобка
8	}	rbrace	Закрывающая фигурная скобка
9	[	lbracket	Открывающая квадратная скобка
10	]	rbracket	Закрывающая квадратная скобка
11	;	semicolon	Точка с запятой
12	,	comma	Запятая
13	:	colon	Двоеточие
Символы операций			
14	=	opassign	Операция присваивания
15	+	opplus	Операция сложения
16	-	opminus	Операция вычитания
17	*	opmult	Операция умножения
18	++	opinc	Операция инкремента
19	==	opeq	Операция равно
20	!=	opne	Операция неравно
21	<	oplt	Операция меньше
22	>	opgt	Операция больше
23	<=	ople	Операция меньше или равно
24	!	opnot	Операция логического отрицания
25	||	opor	Операция логического ИЛИ
26	&&	opand	Операция логического И
Ключевые слова			
27	int	kwint	Тип данных int
28	char	kwchar	Тип данных char
29	if	kwif	Ключевое слово if
30	else	kwelse	Ключевое слово else
31	switch	kwswitch	Ключевое слово switch
32	case	kwcase	Ключевое слово case
33	while	kwwhile	Ключевое слово while
34	for	kwfor	Ключевое слово for
35	return	kwreturn	Ключевое слово return
36	in	kwin	Ключевое слово in
37	out	kwout	Ключевое слово out

Задания на РГР

1. Добавление однострочных комментариев //
2. Добавление многострочных комментариев /* */
3. Добавление директив препроцессора #define, #undef, #ifdef, #else, #endif
4. Добавление оператора взятия остатка от деления %
5. Добавление операции сравнения >=
6. Добавление оператора break в цикл for
7. Добавление оператора break в цикл while
8. Добавление оператора continue в цикл for
9. Добавление оператора continue в цикл while
10. Добавить операторы побитовых сдвигов влево << и вправо >>
11. Добавить оператор префиксного декремента --a
12. Добавить оператор постфиксного декремента a--
13. Добавление цикла **do** [*тело цикла*] **while**(*условие*)
14. Добавление диагностических предупреждений во время компиляции при неявном преобразовании типа int в тип char и операции явного преобразования типов static_cast<тип> (имя)
15. Добавление операции унарного минуса
16. Реализовать ветку default в операторе switch
17. Добавление сокращенного присваивания a += b
18. Реализовать поддержку ключевого слова const и символьных констант
19. Реализовать операцию деления
20. Добавление двумерных массивов
21. Добавление меток и оператора goto