

Правила преобразования программ из МИКРОЛИСПа в C++

- 1) В начале целевой программы записывается директива `#include "mlisp.h"` .
- 2) Вещественные и строковые константы переписываются без изменений.
- 3) Целые константы переводятся в вещественную форму. Например, -1, 1, 003 переписываются как -1. , 1. , 003. .
- 4) Булевские константы `#t` и `#f` заменяются `true` и `false`.
- 5) В идентификаторах
 - а) –(дефис) заменяется на `__`(два подчеркива);
 - б) `!` заменяется на `_E` ;
 - в) `?` заменяется на `_Q` .Например, идентификаторы `good-enough`, `x!a`, `equal?` заменяются `good__enough`, `x_Ea`, `equal_Q` .
- 6) Идентификаторы, совпадающие с ключевыми словами C++, «декорируются» префиксами `__xxx__` . Например, идентификатор `try` заменяется `__xxx__try`. Здесь `xxx` – персональный код студента - инициалы, записанные латиницей.
- 7) Каждому выражению МИКРОЛИСПа соответствует выражение C++.
- 8) Выражениям `cond` и `if` соответствует тернарный условный оператор C++ (*условие ? выражение : выражение*).
- 9) Если в выражении `cond` нет ветви `else`, то эквивалентное выражение C++ завершается, вырабатывая значение `_infinity`, определенное в файле `mlisp.h` .
Например.

```
(cond((< a b) a)  
      ((= a b) b) )
```

переписывается как

```
( a < b ? a  
  : a == b ? b  
  : _infinity )
```
- 10) Составные ветви `cond` переписываются с помощью оператора `“,”`(оператора последовательности).
Например.

```
(cond(< a b) (display "a ") a)
      (else (display "b ") b) )
```

переписывается как

```
( a < b ? display("a "),a
      : (display("b "),b ))
```

11) Определениям процедур и переменных МИКРОЛИСПа соответствуют определения функций и переменных C++.

12) Порядок записи определений сохраняется. Для выполнения этого правила следует использовать предварительные ОБЪЯВЛЕНИЯ всех функций и глобальных переменных в начале программы.

Например.

```
(define (Zero? x) (< (abs x) eps))
(define eps 1e-5)
```

переписывается как

```
bool Zero_Q(double x); //объявление
extern double eps; //объявление
```

```
bool Zero_Q(double x){ //определение
    return abs(x)<eps;}
double eps=1e-5; //определение
```

13) Последнему выражению в теле процедуры соответствует одна единственная инструкция return.

14) Процедурам-предикатам и переменным, чьи имена оканчиваются знаком ? , соответствуют функции и переменные типа bool. Все остальные функции и переменные в C++ имеют тип double.

15) Эквиваленты полных выражений МИКРОЛИСПа, вычисляемых вне определений процедур, помещаются в теле функции main.

16) Форма let имеет два разных применения в МИКРОЛИСПе. Форма let, содержащая определения локальных переменных, называется блоком. Форма let без локальных переменных – «легким let».

16.1) Блок может быть только последним выражением тела процедуры. В C++ он переписывается следующим образом:

МИКРОЛИСП:

```

(define(f x)
  (let( (a (+ x 6))
        (b (* x x))
      )
    (/ a b)
  )
)

```

C++:

```

double f(double x){
  //let
  double a(x+6.),
          b(x*x);
  return a/b;
} //let
} // f

```

NB!! Правила 12 и 16 предполагают, что инициализаторы глобальных и локальных переменных имеют разную форму записи в C++ .

16.2) Легкий let можно использовать в любом контексте, допускающем простое выражение. В C++ оно переписывается с помощью оператора последовательности.

Например.

```

(if(< a b) (let() (display "a ") a)
          (let() (display "b ") b) )

```

переписывается как

```

( a < b ? (display("a "),a )
          : (display("b "),b )
)

```