Руководство пользователя для языка MAK++

Разработчики: Радаева Карина, Трубников Максим

21 апреля 2021 г.

Содержание

1 Введение **2**

2 Структура программы **3**
2.1 Функции и процедуры 3
2.2 main 5

3 Операторы **5**
3.1 Цикл while 5
3.2 Цикл do...while 6
3.3 Цикл for 7
3.4 Цикл elfor 7
3.5 Условный оператор 9
3.6 Оператор прерывания break 9
3.7 Оператор return 10

4 Типы данных **11**
4.1 Типы 11
4.2 Совместимость типов 11
4.3 Совместимость типов и операций 12

5 Примеры программ **12**

1 Введение

Данная документация предназначена для пользователей, пишуших на языке МАК++. Она содержит общие факты о языке и поможет в освоении языка. Если вы хотите более глубоко познакомится с языком можете обратиться к Технической документации. Желаем Удачи!

2 Структура программы

2.1 Функции и процедуры

Программа должна иметь определенную структуру для успешной компиляции. В начале необходимо описать(если необходимо) функции и процедуры.

```
int func() {

}

double func() {

}

boolean func() {

}

string func() {

}
```

Рис. 1: Примеры описания функций

```
def procedure() {

}
```

Рис. 2: Примеры описания процедур

Каждая функция или процедура может иметь неограниченное количество параметров вызова. **Передача параметров производится только по значению!** При передаче аргументов по значению внешний объект, который передается в качестве аргумента в функцию, не может быть изменен в этой функции. В функцию передается само значение этого объекта.

```
int func(double example4_, boolean example1_, string example5_) {

}

double func(int example1_, double example2_, string example3_, boolean example4_) {

}

boolean func(double example4_, boolean example2_) {

}

string func(string example1_, int ex4_) {

}

def procedure(boolean example1_, string example5_) {

}
```

Рис. 3: Примеры передачи параметров

Внутри функций Вы можете выполнять людые действия, допустимые синтаксисом языка. Можете производить арифметические вычисления, выводить что-то, вводить и т.д.

```
int func(double example4_, boolean example1_, string example5_) {
    read(example4_);
}

double func(int example1_, double example2_, string example3_, boolean example4_) {
    write(example3_);
}

boolean func(double example4_, boolean example2_) {
    example4_ = example2_ + 5 * 6 - 6 / (1 + 2);
}

string func(string example1_, int ex4_) {
    ex4_ = 4 + 5;
}

def procedure(boolean example1_, string example5_) {
    read(example5_);
    write(example1_);
}
```

Рис. 4: Примеры функций(процедур)

2.2 main

Главной функцией программы является main(). Именно с неё начинается выполнение программы.

```
main() {

}
```

Рис. 5: main

3 Операторы

Язык содержит в себе основные операторы циклов: while, do...while, for, elfor. И также он содержит условный оператор if...elseif...else. Рассмотрим каждый из них.

3.1 Цикл while

Цикл while имеет следующую структуру:

```
main() {
    while(выражение, совместимое с boolean) {
        // Операторы
    }
}
```

Рис. 6: while

Как видно из примера, в круглых скобках необходимо указать выражение, совместимое с boolean. Пример такого может быть 1==1 или $2 > 1$.
Синтаксис цикла while: while (условие) { оператор }
Вычисляется условие, и если оно истинно, выполняется оператор, затем снова вычисляется условие, и так до тех пор, пока условие не станет ложным. Если условие сразу ложно, оператор не выполняется ни одного раза.

Замечание 1. Условие всегда заключается в скобки.
Замечание 2. Телом цикла может быть составной оператор.

3.2 Цикл do...while

Цикл do...while имеет следующую структуру:

```
main() {
    do {
        // оператор
    while ( условие );
}
```

Рис. 7: do...while

Как видно из примера, в круглых скобках необходимо указать выражение, совместимое с boolean. Пример такого может быть 1==1 или $2 > 1$.
Синтаксис цикла do...while:
do {
оператор
} while(условие);

Выполняется оператор, затем проверяется условие, и если оно истинно, оператор выполняется еще раз, и снова проверяется условие, и т. д. до тех пор, пока условие не станет ложным. Поскольку условие продолжения цикла проверяется в конце тела цикла, оператор тела цикла всегда будет выполнен по крайней мере один раз.

3.3 Цикл for

Цикл for имеет следующую структуру:

```
main() {
    for (expr1_; expr2_; expr3_) {

    }
}
```

Рис. 8: for

Синтаксис цикла for:
for (начальное действие ; условие ; приращение) {
оператор
}
Порядок выполнения цикла for:
1) начальное действие
2) проверка условия
3) оператор (тело цикла)
4) приращение
После пункта 4 проверяется условие 2. Если оно истинно, выполняются пункты 3, 4. Иначе — цикл прекращает свою работу.
Если условие 2) сразу ложно, оператор не выполняется ни одного раза.
Любое из выражений(1,2,3) можно пропустить. Например:

```
main() {
    for (; expr2_; expr3_) {
        // операторы
    }
    for (expr1_;;) {
        // операторы
    }
    for (;;) {
        // операторы
    }
}
```

Рис. 9: Примеры for

3.4 Цикл elfor

Цикл elfor имеет следующую структуру:

```
main() {
    elfor (expr1_; expr2_; expr3_) {
        // операторы
    } else {
        // операторы
    }
}
```

Рис. 10: elfor

Порядок выполнения цикла elfor:
1) начальное действие
2) проверка условия
3) оператор (тело цикла)
4) приращение
После пункта 4 проверяется условие 2. Если оно истинно, выполняются пункты 3, 4. Иначе — цикл прекращает свою работу.
Если условие 2) сразу ложно, оператор не выполняется ни одного раза.
Любое из выражений(1,2,3) можно пропустить. Например:

```
main() {
    elfor (; expr2_; expr3_) {
        // операторы
    } else {
        // операторы
    }

    elfor (expr1_;;) {
        // операторы
    } else {
        // операторы
    }

    elfor (;;) {
        // операторы
    } else {
        // операторы
    }
}
```

Рис. 11: Примеры elfor

В цикле elfor:

- блок else выполняется в том случае, если цикл завершил итерацию

списка

- но else не выполняется, если в цикле был выполнен break

3.5 Условный оператор

Условный оператор имеет следующую структуру:

```
main() {
    if (expr1_) {
        // операторы
    } elseif (expr2_) {
        // операторы
    } elseif (expr3_) {
        // операторы
    } else {
        // операторы
    }
}
```

Рис. 12: Примеры условного оператора

Когда выполнение основной ветки программы доходит до условного оператора if-else, то в зависимости от результата логического выражения в его заголовке выполняются разные блоки кода. Если логическое выражение вернуло true, то выполняется один блок, если false – то другой (начинается со слова elseif или else). После выполнения одного из вложенных блоков кода, ход программы возвращается в основную ветку. Другой вложенный блок не выполняется.

- Условный оператор обязан иметь первую ветку if. Остальные ветки не обязательны.
- Условный оператор может иметь неограниченное количество веток типа elseif.
- Ветка else обязательно является последней. После нее не может быть написана другая ветка условного оператора.

3.6 Оператор прерывания break

Оператор break имеет следующую структуру:

```
main() {
    for (expr1_; expr2_; expr3_) {
        break;
    }
}
```

Рис. 13: Примеры break

```
main() {
    while (expr1_) {
        break;
    }
}
```

Рис. 14: Примеры break

В контексте циклов оператор break используется для завершения работы цикла раньше времени.

3.7 Оператор return

Оператор return имеет следующую структуру:

```
int func() {
    return 1;
}

double func() {
    return 1.0;
}

boolean func() {
    return true;
}

string func() {
    return "text";
}
```

Рис. 15: Примеры return

Оператор return завершает выполнение всей функции, а выполнение продолжается в точке после вызова функции.

4 Типы данных

4.1 Типы

- **int** — целочисленный знаковый тип. Диапозон [-2 147 483 648; 2 147 483 647]
- **boolean** — логический тип. Диапозон true, false или 0, 1.
- **double** — тип данных с плавающей точкой.
- **string** — строковый тип данных.

4.2 Совместимость типов

Тип	int	boolean	double	string
int	+	+	+	-
boolean	+	+	+	-
double	+	+	+	-
string	-	-	-	+

4.3 Совместимость типов и операций

Операция/Тип	int	boolean	double	string
+	+	+	+	-
-	+	+	+	-
*	+	+	+	-
/	+	+	+	-
%	+	+	-	-
==	+	+	+	+
<>	+	+	+	+
<	+	+	+	+
>	+	+	+	+
<=	+	+	+	+
>=	+	+	+	+
++	+	+	+	-
–	+	+	+	-
||	+	+	+	+
&&	+	+	+	+
«	+	+	-	-
»	+	+	-	-
		+	+	-	-
XOR	+	+	-	-
=	+	+	+	+
+=	+	+	+	+
-=	+	+	+	-
*=	-	-	-	-
/=	+	+	+	-
%=	+	+	-	-

5 Примеры программ

- **Функция Аккермана**

```
int akkerman(int m, int n) {
        if (m == 0) {
                return n + 1;
        } elseif (m > 0 && n == 0) {
                int fg = m - 1;
                return akkerman(fg, 1);
        } elseif (m > 0 && n > 0) {
                int fg = m - 1;
                int kl = n - 1;
                int sd = akkerman(m, kl);
                return akkerman(fg, sd);
        }
}

main() {
        int m;
```

```
        int n;
        write("Akkerman function", "\n");
        write("input m: ");
        read(m);
        write("\n", "input n: ");
        read(n);
        write("\n");
        int res = akkerman(m, n);
        write(res);
}
```