Основы программирования. Введение

Лекции по курсу

Основы программирования

семестр 1, ФИИТ

Михалкович С.С.

Курс ОП + ЯП

Ресурсы

forum.mmcs.sfedu.ru – топик поддержки курса
it.mmcs.sfedu.ru – сайт направления ФИИТ
edu.mmcs.sfedu.ru – среда Moodle поддержки практики
pascalabc.net – сайт системы программирования PascalABC.NET

Семестры

Семестр 1. Основы программирования. Язык PascalABC.NET
Семестр 2. Языки программирования. Язык C#
Семестр 3. Языки программирования. Язык C++

Алгоритмы

Алгоритм – последовательность действий, приводящая к решению поставленной задачи

Свойства алгоритма:

- Дискретность (разбит на шаги)
- Детерминированность (один результат для одних и тех же данных)
- Конечность
- Массовость

Действия алгоритма – это команды некоторого **Исполнителя**. Исполнитель определяет набор команд, в терминах которых должен записываться алгоритм

В сфере ИТ исполнителем алгоритма является **центральный процессор** *компьютера*. Он имеет около 200 машинных команд, в которые переводится любая программа.

Более абстрактный исполнитель – класс с набором методов, в терминах которых решается задача.

Пример алгоритма

Задача.

```
Дано: x, y, z
Найти: max
```

Алгоритм 1. (*словесное описание*)

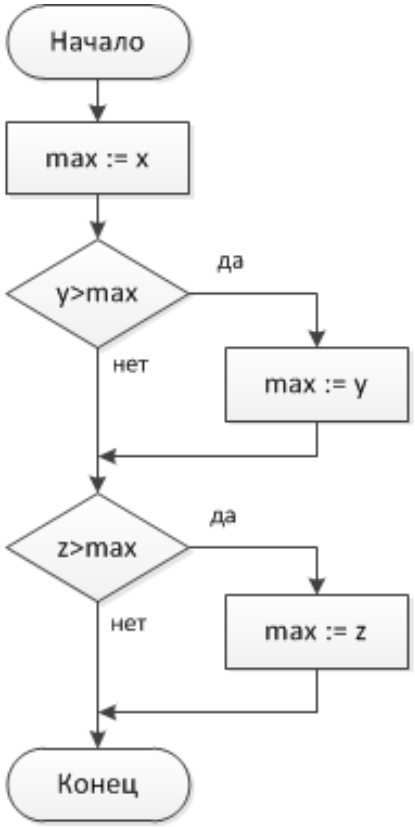
```
Если x больше или равен y и z, то максимум – это x
Если y больше или равен x и z, то максимум – это y
Если z больше или равен x и y, то максимум – это z
```

Алгоритм 2. (*псевдокод*)

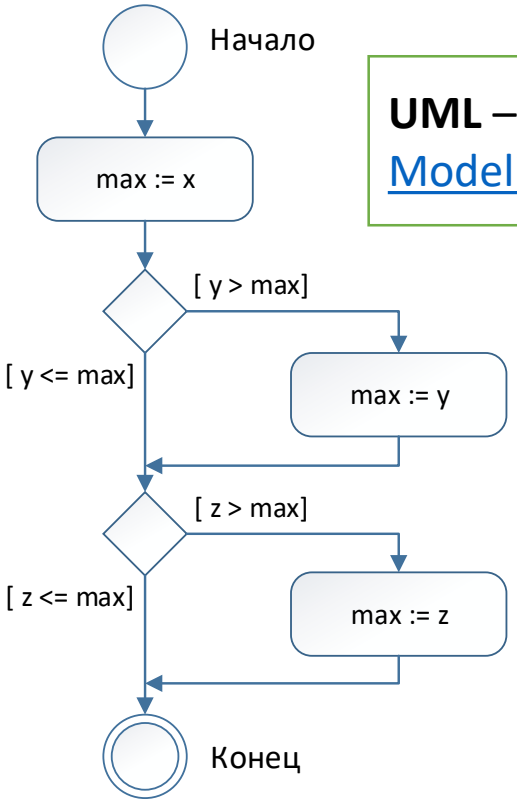
```
max := x
Если y > max то max := y
Если z > max то max := z
```

Способы описания алгоритмов

1. Словесный
2. Псевдокод
3. Блок-схемы



4. Диаграмма деятельности UML (activity diagram)



UML – Unified Modeling Language

5. Язык программирования (ЯП)

Описание алгоритмов в терминах методов классов

Шаги алгоритма – это операторы языка программирования.

В языке программирования могут быть высокоуровневые объекты и классы, содержащие ряд методов.

Методы классов и стандартные подпрограммы (функции) также образуют команды некоторого исполнителя, в терминах которого может быть решена данная задача.

Пример. Дана последовательность целых.

Найти два первых минимальных элемента.

Какие возможны решения?

```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
...
```

Решения задачи о двух минимумах

Решение 1. Отсортировать по возрастанию, удалить дубли.
Взять первые два значения:

Функция, создающая последовательность

```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
Sq := Sq.Order.Distinct;
var (Min,Min2) := (Sq.ElementAt(0),Sq.ElementAt(1));
```

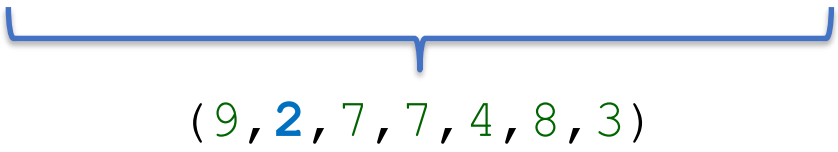
Решения задачи о двух минимумах

Решение 1. Отсортировать по возрастанию, удалить дубли.
Взять первые два значения:

```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
Sq := Sq.Order.Distinct;
var (Min,Min2) := (Sq.ElementAt(0),Sq.ElementAt(1));
```

Решение 2. Найти первый минимум.
Отфильтровать последовательность, удалив первый минимум.
В отфильтрованной последовательности найти минимум

```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
var Min := Sq.Min;
var Min2 := Sq.Where(elem -> elem <> Min).Min;
```



Решения задачи о двух минимумах

Решение 1. Отсортировать по возрастанию, удалить дубли.
Взять первые два значения:

```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
Sq := Sq.Order.Distinct;
var (Min,Min2) := (Sq.ElementAt(0),Sq.ElementAt(1));
```

Решение 2. Найти первый минимум.
Отфильтровать последовательность, удалив первый минимум.
В отфильтрованной последовательности найти минимум

```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
var Min := Sq.Min;
var Min2 := Sq.Where(elem -> elem <> Min).Min;
```

(9,2,7,7,4,8,3)

Это двухпроходный алгоритм. За него ругали на ЕГЭ :)

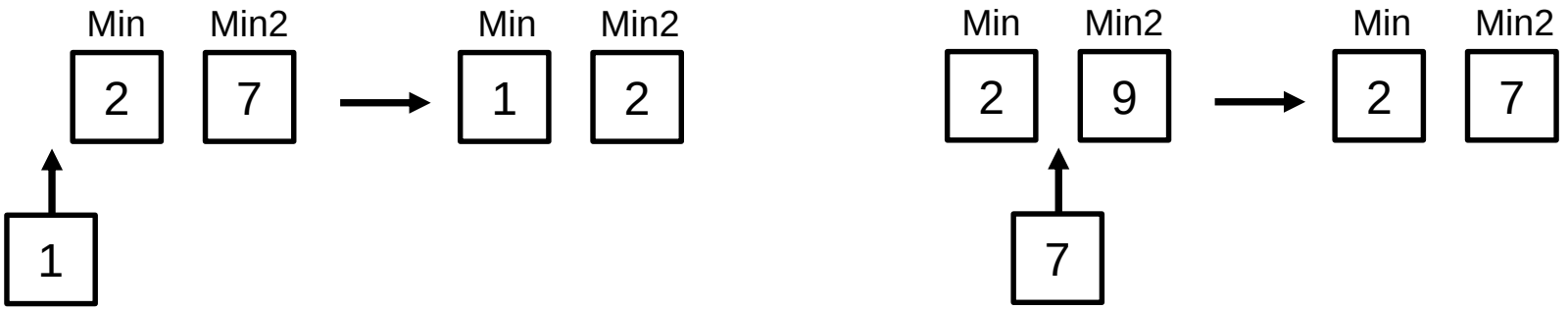
Решения задачи о двух минимумах

Решение 3. Как хотят в ЕГЭ.

Для каждого элемента elem последовательности:

```
    если elem < Min то
        (Min, Min2) := (elem, Min)
    иначе если elem внутри (Min, Min2) то
        Min2 := elem
```

Пояснение.



Вопрос. Чем инициализировать Min и Min2 в начале?

Решения задачи о двух минимумах

Решение 3. Как хотят в ЕГЭ.

Для каждого элемента elem последовательности:

если elem < Min то

(Min, Min2) := (elem, Min)

иначе если elem внутри (Min, Min2) то

Min2 := elem

```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
var (Min,Min2) := (integer.MaxValue, integer.MaxValue);
foreach var elem in Sq do
  if elem < Min then
    (Min, Min2) := (elem, Min)
  else if (elem > Min) and (elem < Min2) then
    Min2 := elem;
```

Решения задачи о двух минимумах

Решение 3. Как хотят в ЕГЭ.

Для каждого элемента elem последовательности:

если elem < Min то

(Min, Min2) := (elem, Min)

иначе если elem внутри (Min, Min2) то

Min2 := elem

Вопрос. Чем инициализировать Min и Min2 в начале?

```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
var (Min,Min2) := (integer.MaxValue, integer.MaxValue);
foreach var elem in Sq do
  if elem < Min then
    (Min, Min2) := (elem, Min)
  else if (elem > Min) and (elem < Min2) then
    Min2 := elem;
```

Зато однопроходный

Эквивалентные алгоритмы

Алгоритмы называются **эквивалентными** если

- множества их исходных данных совпадают
- на одних и тех же данных они выдают одинаковый результат

Вопрос. Какой из эквивалентных алгоритмов лучше?

Вопрос. Каковы критерии сравнения алгоритмов и программ?

Критерии сравнения программ

1. Скорость работы
2. Объём требуемой памяти

Критерии сравнения программ

0. **Правильность**
1. Скорость работы
2. Объём требуемой памяти

Критерии сравнения программ

0. **Правильность**
1. Скорость работы
2. Объём требуемой памяти
3. Читаемость (простота понимания кода)
4. Простота написания кода
5. Скорость разработки
6. Модифицируемость
7. Масштабируемость (способность работать на бОльших объёмах данных)
8. Безопасность
9. Стоимость
10. Переносимость на другие платформы

Вопрос. Какой критерий более важен?

Задача о двух минимумах

Какое решение лучше? Почему?

1.
```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
Sq := Sq.Order.Distinct;
var (Min,Min2) := (Sq.ElementAt(0),Sq.ElementAt(1));
```

2.
```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
var Min := Sq.Min;
var Min2 := Sq.Where(elem -> elem <> Min).Min;
```

3.
```
var Sq := Seq(9,2,7,7,4,1,8,1,3);
var (Min,Min2) := (integer.MaxValue, integer.MaxValue);
foreach var elem in Sq do
  if elem < Min then
    (Min, Min2) := (elem, Min)
  else if (elem > Min) and (elem < Min2) then
    Min2 := elem;
```

Компиляторы и интерпретаторы

Компилятор – программа, переводящая текст программы на языке высокого уровня в машинные коды.

На этапе компиляции текст программы проверяется на ошибки: лексические, синтаксические, семантические (примеры – позже)

Откомпилированная программа заведомо не содержит таких ошибок. Однако может произойти **ошибка времени выполнения**.

Интерпретатор – программа, непосредственно выполняющая программу на языке программирования команда за командой.

Недостатки:

1. ошибки проверяются в момент выполнения
2. низкая скорость выполнения программы

Достоинства:

1. существенно более простой, чем компилятор. Его можно с легкостью переносить на новую платформу

Язык программирования PascalABC.NET

Включает в себя:

1. Базовый Паскаль
2. Delphi Паскаль
3. .NET расширения
4. Собственные расширения

Ориентирован на:

1. Современный процесс обучения программированию
2. Компактную запись кода
3. Современную платформу .NET

Итог после 1 семестра:

1. Изучены основные конструкции и концепции программирования
2. Обеспечен бесшовный переход к языку C#

Простейшая программа

Раздел описаний – до begin

Описания – внутри begin, автоопределение типа переменной типом инициализирующего значения

Классический Паскаль

```
(*Площадь и периметр прямоугольника*)
var a,b,S,P: real;

begin
  Write('Введите a,b: ');
  Readln(a,b);
  S := a*b;
  P := 2*(a+b);
  Writeln('Площадь = ', S);
  Writeln('Периметр = ', P);
end.
```

PascalABC.NET

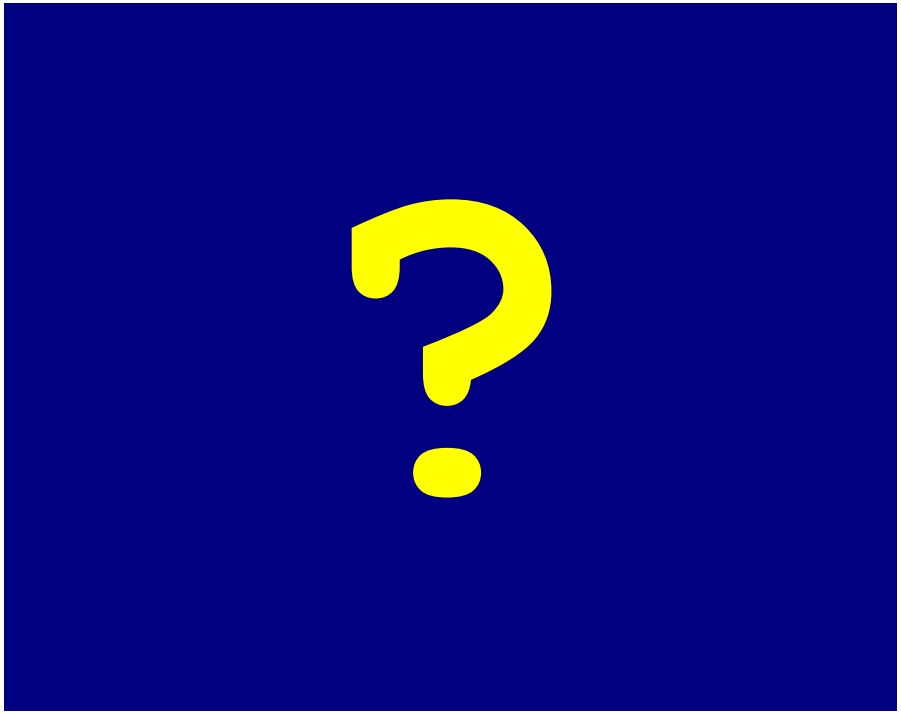
```
// Площадь и периметр прямоугольника
begin
  var a := ReadReal('Введите a:');
  var b := ReadReal('Введите b:');
  var (S,P) := (a*b,2*(a+b));
  Println('Площадь =', S);
  Println('Периметр =', P);
end.
```

Кортежи

Распаковка кортежа в переменные

Простейшая программа 2: вычисление n!

Классический Паскаль



PascalABC.NET

```
begin
  var n := ReadInteger;
  var p: BigInteger := 1;
  for var i:=1 to n do
    p *= i;
  Println($'{n}!={p}');
end.
```

Интерполированные строки

Типы данных

Основные типы:

```
integer  real  BigInteger
boolean  char  string
```

Тип данных определяет:

1. Множество значений данного типа
2. Набор операций над значениями этого типа
3. То, как хранятся значения данного типа в памяти

Перед использованием все переменные должны быть описаны. При описании указывается **имя** и **тип:**

```
var i: integer;
var r: real;
```

В PascalABC.NET можно описывать переменные с **автовыведением** типа.

```
var i := 666;
var r := 3.14;
```

Автовыведение типа

В PascalABC.NET можно описывать переменные с **автовыведением** типа:

```
var i := 666;
var r := 3.14;
var r1 := i / 2; // какой тип?
var s := 'ABC';
var c := 'A';
```

Множественное описание переменных с **автовыведением** типа:

```
var (a,b) := (3,5);
var (r,s) := (3.14, 'ABC');
```

Присваивание

Базовый оператор

```
a := b + 3.14;
```

1. Подставляет вместо b ее значение
2. Вычисляет значение выражения в правой части
3. Если тип выражения в правой части **совместим по присваиванию** с типом переменной в правой части, то в ячейку a записывается новое значение

Типы **совместимы по присваиванию** если:

1. они совпадают
2. real := integer
3. string := char
4. BigInteger := integer

Пример ошибки:

```
var i: integer := 1;
i := i + 3.14;
```

Присваивание - примеры

Задача. Поменять местами значения переменных a и b

Решение 1. С помощью временной переменной

```
var t := a;
a := b;
b := t;
```

Решение 2. Заумное. Какие у него недостатки?

```
a := a + b;
b := a - b;
a := a - b;
```

Решение 3. С помощью стандартной процедуры

```
Swap(a, b);
```

Решение 4. С помощью **множественного присваивания**

```
(a, b) := (b, a);
```

Вопрос. Какое решение лучше?

Присваивание и временные переменные

Задача. Вычислить a^{16}

Решение 1. В лоб. 15 умножений

```
var res := a*a*a*a * a*a*a*a * a*a*a*a * a*a*a*aё;
```

Решение 2. Введением временных переменных. 4 умножения

```
var t := a * a;
t := t * t;
t := t * t;
var res := t * t;
```

Решение 3. С помощью операции возведения в степень

```
var res := a ** 16;
```

Задача. Вычислить a^{15} за минимальное число умножений

```
var t := a * a;
t := t * t;
t := t * a; // a^5
var res := t * t * t;
```

Задача. Вычислить a^n за минимальное число умножений (на дом)

Ввод данных

Традиционный ввод

```
var a,b: integer;
Writeln('Введите a,b: '); // приглашение к вводу
Readln(a,b);
```

Ввод, совмещенный с описанием и автовыведением типа

```
var a := ReadInteger;
var b := ReadInteger;
```

Ввод нескольких переменных, совмещенный с описанием

```
var (a,b) := ReadReal2;
```

Ввод, совмещенный с описанием и приглашением к вводу

```
var (a,b) := ReadReal2('Введите a,b:');
```

Безопасный ввод

Ошибка выполнения при вводе

```
var a: real;
Read(a); // Вводим a=3.14
```

Program1.pas(3) : Ошибка времени выполнения: Входная строка имела неверный формат.

Ввод с защитой от неверного ввода

```
var a: real;
var b: boolean := TryRead(a);
```

Если при вводе – ошибка, то программа не завершается и в переменной `b` – значение `False`

Если ошибки нет, то `b=True` и в переменной `a` – введенное значение

Вывод

Print vs Write

```
Writeln(1,2,3);          // 123
Writeln(1,' ',2,' ',3); // 1 2 3
Println(1,2,3);          // 1 2 3
```

Вывод выражений и поясняющих строк

```
Println(a,'+',b,'=',a+b); // 2 + 3 = 5
```

Вывод с использованием форматной строки

```
WritelnFormat('{0}+{1}={2}',a,b,a+b); // 2 + 3 = 5
```

Форматирование с использованием форматной строки

```
WritelnFormat('{0,5}',5);        // ____5
WritelnFormat('{0,9:f3}',3.14); // ____3.140
```

Вывод с использованием интерполированной строки

```
Println($'{a}+{b}={a+b}');
```

Ошибки в программе

Лексическая ошибка

```
   var % := 1;
```

```
Program1.pas(2) : Неожиданный символ '%'
```

Синтаксическая ошибка

```
   var a := 1
   var b := 2;
```

```
Program1.pas(3) : Встречено 'var', а ожидалось ';'
```

Семантическая ошибка

```
   var a: integer;
   var r: real;
   a := r;
```

```
Program1.pas(4) : Нельзя преобразовать тип real к integer
```

Ошибка времени выполнения

```
   var a := ReadInteger; // Вводим hello
```

```
Program1.pas(2) : Ошибка времени выполнения: Входная строка
имела неверный формат.
```

Стандартные арифметические функции

Наиболее часто используемые арифметические функции

```
Sqrt(x)  – квадратный корень
Abs(x)   - модуль
Min(x,y) – минимум
Max(x,y) - максимум
Round(x) – округление вещественного
Trunc(x) – отбрасывание дробной части вещественного
Random(a,b)  – случайное в диапазоне [a,b]
Random2(a,b) – пара случайных в диапазоне [a,b]
```

Методы

```
x.Sqrt
x.Round
x.Trunc

x.Sqrt.Round
```

Q & A