

# ПОДГОТОВИТЕЛЬНЫЙ ЭТАП ИЗУЧЕНИЯ АЛГОРИТМОВ ОБРАБОТКИ МАССИВОВ

Ф.С. ШУМЧИК

*Учреждение образования “Белорусский государственный университет  
информатики и радиоэлектроники” филиал “Минский радиотехнический  
колледж”*

**Аннотация.** Тема «Алгоритмы обработки массивов» в учебном предмете «Информатика» является обобщающей по основам алгоритмизации и программирования. Поэтому важно на первом занятии пройти повторение базовых понятий алгоритмизации, освежить в памяти основные команды и операции, что даст возможность успешно освоить новый материал по обработке массивов.

В системе среднего специального образования учащиеся 1 курса, обучающиеся на основе общего базового образования, знакомятся с алгоритмами обработки массивов. В соответствии с учебной программой по учебному предмету «Информатика» эта последняя тема по основам алгоритмизации и программирования в школьном курсе по информатике. Учитывая важность освоения темы учащимися специальностей профиля информационно-коммуникативных технологий, а также небольшое количество отводимых на изучение учебной программой учебных часов, необходимо построить занятия таким образом, чтобы за короткое время обобщить знания учащихся по основам алгоритмизации и программирования, что даст возможность успешно освоить новый материал по обработке массивов.

На первом занятии следует с обучающимися пройти повторение базовых понятий алгоритмизации, освежить в памяти основные команды и операции.

Учащиеся знакомы с командами (процедурами) вывода `write`, `writeln`, ввода `read`, `readln` (здесь и далее обязательная демонстрация примеров).

Основная часть любой программы состоит из команд, которые обрабатывают данные. Одной из таких команд является команда присваивания. С помощью команды присваивания можно задавать значения переменных (вводить исходные данные), вычислять значения арифметических выражений и результат записывать в переменные.

Константы описываются перед программным блоком `begin ... end` в так называемом разделе описания констант, начинающемся служебным словом `const`.

Кроме констант, в программе могут присутствовать величины другого рода, значение которых может изменяться в процессе выполнения программы. Такие величины называются переменными. Каждой переменной назначается имя (идентификатор), с помощью которого затем оперируем с ней. Как и константа, переменная имеет тип.

Для описания переменных используется команда `var` (сокр. от англ. *variable* – переменная). Формат записи команды:

```
var <имя переменной>: <тип>; // var a : integer;
```

Чтобы использовать какую-либо переменную, ее нужно описать. Описание переменных выполняется до начала программы (команды *begin*).

Все переменные, которые используются в программе, должны быть описаны в разделе *var*. Если данные не изменяются в процессе работы программы, то они могут быть описаны как константы в разделе *const*.

При описании переменной можно присвоить ей начальное значение (это называется инициализацией), но лишь одной для каждого *var*:

*var* Имя переменной:= значение; (например, *var:=5*);

Конструкция *:=* имеет название знака операции присваивания (оператор присваивания) и обозначает следующее: необходимо вычислить значение выражения, помещенного справа от знака операции присваивания и поместить его в переменную, имя которой указано слева от этого знака.

В алгоритмизации определены операторы присваивания со следующими значками: *+=*; *-=*; *\*=*; */=*. Они позволяют изменить значение переменной. Например: *a\*= 5*; (увеличить *a* в 5 раз); *b -= 6*; (уменьшить *b* на 6).

Учащиеся умеют работать с переменными целого типа: *+* (сложение), *-* (вычитание), *\** (умножение), *div* (целочисленное деление), *mod* (деление по модулю).

Целочисленные переменные можно обрабатывать с помощью стандартных функций *abs(x)* и *sqr(x)*, где аргумент *x* – это переменная либо выражение целого типа. Функция *abs(x)* вычисляет абсолютное значение переменной *x*, функция *sqr(x)* возводит в квадрат значение переменной *x*. При этом важно напомнить, что аргументы функции всегда пишутся в скобках.

Для типа данных *integer* определены следующие операции (знаки математических действий): сложение (*+*), вычитание (*-*), умножение (*\**), целочисленного деления (*div*), нахождения остатка (*mod*).

Следует помнить, что операция деления (*/*) не используется при вычислениях с данными типа *integer*. Для данных этого типа используются операции *div* и *mod* [1, с. 31].

Необходимо обратить внимание на переменные вещественного типа, над которыми можно выполнять арифметические операции сложения, вычитания, умножения, обычного деления (*/*), а также обрабатывать их с помощью стандартных функций *abs(x)*, *sqr(x)*, *trunc(x)*, *round(x)*, *frac(x)*. Учащиеся раскрывают значение каждой функции (например, *trunc(x)* вычисляет целую часть вещественного числа *x*; *round(x)* округляет вещественное число *x* до целого; *frac(x)* вычисляет дробную часть вещественного числа *x*; *abs(x)* вычисляет модуль числа *x*; *sqr(x)* вычисляет квадрат числа *x*).

Для типа данных *string* допустимы строковые функции и процедуры (*length* и др.). Для строк определена операция сложения (конкатенация). Обозначается операция знаком «*+*». В результате сложения двух строк получается новая строка, в которой после символов первой строки будут записаны символы второй строки.

Важно напомнить, что в программировании целая часть десятичной дроби отделяется от дробной части точкой. На нуль делить нельзя.

Обратим внимание на алгоритмическую конструкцию повторения: `for <параметр цикла>:=<начальное значение> to <конечное значение> do <оператор>` (`for i:=N1 to N2 do <оператор>`) или `for <параметр цикла>:=<начальное значение> downto <конечное значение> do <оператор>` (`for i:=N1 downto N2 do <оператор>`). В этих записях `for.. do` – заголовок цикла, `<оператор>` – тело цикла, `i` – параметр цикла.

Формат команды:

```
for var i:= N1 to N2 do
```

```
begin
```

```
  тело цикла;
```

```
end;
```

или:

```
for var i:= N2 downto N1 do
```

```
begin
```

```
  тело цикла;
```

```
end;
```

Ключевое слово `var` может быть опущено, тогда переменная `i` должна быть описана (как `integer`) в разделе описания `var` до начала программы.

Необходимо помнить следующее: если в заголовке оператора `for..to..do` начальное значение параметра цикла больше конечного значения, то тело цикла не выполнится ни разу; если в заголовке оператора `for..downto..do` начальное значение параметра цикла меньше конечного значения, то тело цикла также не выполнится ни разу.

Переходим к алгоритмической конструкции ветвления.

Одна из таких команд реализуется с помощью условного оператора `if`. В общем виде этот оператор выглядит так:

```
if <условие> then <оператор _ 1> else <оператор _ 2>;
```

Оператор `if` анализирует некоторое условие. Если условие верно (истинно), то выполняется оператор `_ 1`, иначе (если условие неверно, ложно) выполняется оператор `_ 2`. Оператор `_ 1` и оператор `_ 2` называются ветвями программы.

Необходимо напомнить о построении составных условий для проверки выполнения двух и более простых условий ( $2 < a < 8$ ). Составные условия строятся из простых с помощью следующих логических операций: `and` – логическое «и»; `or` – логическое «или»; `not` – логическое отрицание: `(a>2) and (a<7)`, `(x>8) or (y<10)`. Простые условия обязательно заключаются в круглые скобки, так как логические операции имеют приоритет перед операциями сравнения.

Переходим к циклу с предусловием – оператор `while`, который имеет следующий вид: `while <условие> do <оператор>;`. Вспоминаем, в чем отличие от оператора цикла с параметром `for`? Цикл с параметром `for` удобен, когда число повторений (итераций) действий заранее известно. В случае, когда число повторений тела цикла заранее неизвестно и определяется только в ходе выполнения цикла, применяем цикл с предусловием. Цикл с предусловием – это цикл, который повторяется до тех пор, пока условие истинно. Если условие истинно, выполняется тело цикла и снова проверяется условие, если условие

становится ложным, происходит выход из цикла. Если условие сразу оказывается ложным, оператор `while` не выполняется ни разу!

Любой цикл `for` можно заменить циклом `while`. В большинстве случаев удобнее использовать цикл `while`.

В программировании для получения случайного числа используют функцию `random`. Способ записи функции:

`random (a, b);` – возвращает случайное целое в диапазоне от `a` до `b`;

Функция `clRandom` позволяет задать случайный цвет.

При решении задачи на вычисление суммы, произведения элементов массива обращаем внимание на начальное значение для суммы и произведения:

`sum := 0;`

`pr := 1` (для произведения начальное значение 1, так как на 0 не умножают).

Имеется еще одна мощная разновидность описания переменных, объединенная с присваиванием начального значения и автовыведением типов. Она основана на так называемом кортежном присваивании:

`var (Имя 1, Имя 2, ... Имя N) := (Значение 1, Значение 2, ... Значение N);`

Здесь переменная `Имя 1` получает `Значение 1` (и соответствующий тип), `Имя 2` получает значение 2 и т.д.: `var (a, b, c, i) := (132, -58, 0, 1);`.

Конструкция, записанная в круглых скобках, называется кортежем, отсюда и происходит название этого вида множественного присваивания.

Необходимо обратить внимание, что указывать тип переменных в кортежном списке не надо [2, с. 43].

Таким образом, повторение на первом занятии таких базовых понятий алгоритмизации, как команда ввод-вывод, присваивание, операторы присваивания, алгоритмическая конструкция повторения, ветвление, условный оператор `if`, цикл с параметром `for`, цикл с предусловием `while`, умение работать с переменными, константами, стандартными функциями позволяет организовать эффективную работу учащихся по усвоению важной темы, связанной с алгоритмами обработки массивов.

### **Список использованных источников**

1. Котов, В.М. Информатика : учебное пособие / В.М. Котов, А.И. Лапо, Ю.А. Быкадоров, Е.Н. Войтехович. – Минск, 2020. – 111 с.
2. Осипов, А.В. PascalABC.NET: Введение в современное программирование / А.В. Осипов. – Ростов-на-Дону: Издательство Южного федерального университета, 2019. – 572 с.