ПЯВУ. Лекция 6.

Основы программирования.

А.М. Задорожный

Вопросы для повторения

1. Назовите 4 обязательных характеристики переменной?
2. Что определяет типа данных?
3. Зачем определены 2 числовых типа данных: int и double?
4. Какие операции можно выполнять над строками?
5. Как происходит сравнение строк?
6. Опишите “время жизни” переменной.

Содержание

1. Массивы. Одномерные массивы
2. Понятие алгоритма, запись алгоритма
3. Примеры простейших алгоритмов
   a) Поиск наибольшего/наименьшего
   b) Поиск условного наибольшего/наименьшего
4. Дополнительные операторы управления циклом (break и continue)
5. Массивы. Многомерные массивы

Массивы

- Важнейшая структура данных

Массив – совокупность элементов данных одного типа, имеющих одно имя, доступ к которым выполняется по номеру элемента.

Массив – новый тип данных. Абстрактный. Контейнер.

Примеры

```
int [] v = new int [16];        // 16 целых

double [] a = new double [4]; // 4 double
```

Массивы могут быть образованы из данных любого определенного ранее типа.

В общем виде:

<тип> [] <имя массива> = new <тип>[<количество>];

Доступ к элементам массива

```
int [] v = new int [16];


int x = v[3];     // чтение элемента массива
v[1] = x;         // изменение элемента
```

Нумерация элементов массива

У объекта–массива имеется свойство **Length**.

```
int n = v.Length; // количество элементов
```

Нумерация элементов всегда начинается с 0;

```
//Какой тип данных напоминает?
```

Перебор элементов

```
double [] v = new double [16];
//Заполнение массива
for(int i = 0; i < v.Length; i++)
{
    v[i] = …
}
```

Инициализация массива

Массивы можно инициализировать как и обычные переменные.

int [] a = **new** int[3] **{1, 2, 3}**;

- Для числовых типов, если массив не инициализирован, все его элементы равны 0;
- Для логических – false;
- Для строковых – пустой строке.

Применения массивов

Вектора в математике в программировании задаются массивами.

Задача. Найти скалярное произведение векторов a(0,1,1) и b(1,1,0)

```
double [] a = new double[3]{0,1,1};
double [] b = new double[3]{1,1,0};

double dp = 0;
for(int i = 0; i < a.Length; i++)
{
    dp += a[i] * b[i];
}
//Здесь в dp содержится скалярное произведение векторов
```

Контрольные вопросы

1. Что такое массив?
2. Могут ли элементы одного массива иметь разные типы?
3. Какой номер имеет последний элемент массива v?
4. Чему равен элемент массива v с номером 1?

```
int [] v = new int [3] {1, 2, 3};
```

5. Что общего между массивом символов и строкой?
6. В чем различие между массивом символов и строкой?

Алгоритм

Алгоритм – конечная последовательность выполнимых действий, которая приводит к решению задачи.

Поваренная книга - пример сборника алгоритмов: имеется задача, исходные данные, указана последовательность действий для достижения цели.

Текстовая запись алгоритма

Задача. Найти скалярное произведение двух векторов *одинаковой размерности*.

Обозначим вектора имени a и b, а результат dp.

1. Занулить аккумулятор dp (dp = 0)
2. Цикл по элементам вектора a. (i-тый компонент)
    1. Увеличить аккумулятор dp на произведение i-тых (текущих) компонентов векторов a и b.
3. Конец цикла 2

В dp содержится скалярное произведение a и b.

Поиск наибольшего/наименьшего

Задача. Найти значение наибольшего элемента массива.

Обозначим наибольшее значение aMax, а массив a.

1. Присвоим aMax значение первого элемента массива;
2. Цикл по элементам массива начиная со второго
    1. Очередной элемент массива a_i сравниваем с aMax
        1. Если a_i больше aMax, то присвоить значение a_i в aMax
3. Конец цикла 2

В aMax содержится значение наибольшего элемента массива a.

Что изменится, если нужно найти наименьший?

Программирование алгоритма

```
//Присвоим aMax значение первого элемента массива
double aMax = a[0];
//Цикл по элементам массива начиная со второго
for(int i = 1; i < a.Length; i++)
{
    //Очередной элемент массива a_i сравниваем с aMax
    if(a[i] > aMax)
    {
        //Если a_i больше aMax, то присвоить значение a_i в aMax
        aMax = a[i];
    }
//Конец цикла 2
}
```

Поиск наибольшего/наименьшего

Задача. Найти номер первого наибольшего элемента массива.

Обозначим номер наибольшего элемента iMax, а массив a.

1. Присвоим iMax значение 0;
2. Цикл по элементам массива начиная со второго
    1. Очередной элемент массива a_i сравниваем с a_{iMax}
        1. Если a_i больше a_{iMax}, то присвоить значение i в iMax
3. Конец цикла 2

В iMax содержится номер первого наибольшего элемента массива a.

Как изменится, если нужен номер последнего наибольшего?

Программирование алгоритма

```
//Присвоим iMax значение 0
int iMax = 0;
//Цикл по элементам массива начиная со второго
for(int i = 1; i < a.Length; i++)
{
    //Очередной элемент массива ai сравниваем с aiMax
    if(a[i] > a[iMax])
    {
    //Если ai больше aiMax, то присвоить значение i в iMax
        iMax = i;
    }
//Конец цикла 2
}
```

Усложнение задачи

Задача. Найти значение наибольшего элемента массива целых, среди элементов, делящихся на 3.

Исходная задача решалась так:

```
int aMax = a[0];
for(int i = 1; i < a.Length; i++)
{
    if(a[i] > aMax)
        aMax = a[i];
}
```

Как ее изменить?

Укрупнение алгоритма

1. Найти первый элемент массива, который делится на 3 и присвоить его в aMax.

2. Применить алгоритм “поиск наибольшего” к оставшимся элементам массива, пропуская те, которые не делятся на 3.

3. Что произойдет, если у в массиве не будет элементов, которые делятся на 3?

Уточнение алгоритма

Будем искать НОМЕР элемента

1. Поместить в iMax значение -1
2. Найти номер первого элемента массива, который делится на 3 и присвоить его в iMax.
3. Если в массиве нет будет элементов, которые делятся на 3, то iMax будет равно -1. На этом решение задачи завершено. Искомого элемента нет!
4. Применить алгоритм “поиск наибольшего” к последующим (iMax) элементам массива, пропуская те, которые не делятся на 3.
5. Если в массиве не будет элементов, которые делятся на 3, то iMax будет равно -1. Если такие элементы есть, то iMax равно номеру наибольшего элемента

Реализация алгоритма

```
int iMax = -1;
for(int i = 0; i < a.Length; i++)
	if(a[i] % 3 == 0)
	{
		iMax = i;
		break;
	}
//Здесь известен номер первого элемента, который делится на 3
if(iMax  >=  0)
{
	for(int i = iMax + 1; i < a.Length; i++)
	if(a[i] % 3 == 0 && a[i] > a[iMax])
		iMax = i;
}
// Здесь iMax или равно -1 или равно номеру наибольшего элемента массива, среди тех, которые делятся на 3.
```

Дополнительные операторы управления циклом

break – прекращает выполнение цикла

continue – переход к следующей итерации

```
for(int i = 0; i < 100; i++)
{
    if(i%5 == 0) continue;
    …
}
```

Контрольные вопросы

1. Что такое алгоритм?
2. Что значит: “Задача не имеет алгоритмического решения”?
3. Что означает фраза: “Степень детализации алгоритма”?
4. Для чего применяется ключевое слово **break**?
5. Для чего применяется ключевое слово **continue**?
6. Как будет выглядеть алгоритм задачи:

“Найти среднее арифметическое элементов массива”?

7. В чем отличие алгоритма и программы, которые вы разрабатываете для реализации алгоритма?

Многомерные массивы

Можно создавать массивы более чем из одной размерности.

int [,] m = **new** int[**3, 3**] **{{1, 2, 3}, {4,5,6}, {7,8,9}}**;
У такого массива свойство Length говорит об общем количестве элементов. (в примере m.Length == 9)

Чтобы узнать размер массива по одному из измерений нужно пользоваться методом **GetLength(n)**, где n – целое число, номер измерения.

Нумеруются измерения с 0, т.е. m.GetLength(0) вернет количество строк матрицы m, а m.GetLength(1) – количество столбцов.

Элемент такого массива определяется 2-мя индексами

m[1,2] = …
int t = m[0,2]

Массив массивов

Элементами массива могут быть и массивы.

```
int [][] a = new int [3][];
```

Пользоваться таким массивом еще нельзя. Нужно задать каждый из его элементов.

```
for(int i = 0; I < a.Length; i++)
    a[i] = new int[3];
```

Теперь задана матрица, которая состоит из 3 строк по 3 элемента в каждой.

Пример использования массива массивов

У массива массивов каждый элемент-массив может иметь свою длину. Вот как можно выяснить длину каждого элемента.

```
for(int i = 0; i < a.Length; i++)
    Console.WriteLine("Элемент {0} имеет длину {1}.",
                                    i,    a[i].Length);
for(int i = 0; i < a.Length; i++)
      for(int j = 0; j < a[i].Length; j++)
            a[i][j] = i+j;
```

Применения многомерных массивов

Матрицы в математике задаются двумерными массивами.
Задача. Найти сумму 2-х матриц одинаковой размерности.

```
double [,] a = new double[3,3]{{0,1,1}, {1,1,1}, {0,1,1}};
double [,] b = new double[3,3]{{1,1,0}, {2,0,0}, {3,1, 0}};

double [,] ab = new double[a.GetLength(0), a.GetLength(1)];

for(int i = 0; i < a.GetLength(0); i++)
      for(int j = 0; j < a.GetLength(1); j++)
            ab[i, j] = a[i, j] + b[i, j];

//Здесь в ab содержится сумма двух матриц a и b.
```

Контрольные вопросы

1. Как можно задать двумерный массив?
2. Чем двумерный массив отличается от массива массивов?
3. О чем говорит в многомерном массиве свойство Length?
4. Как узнать количество элементов многомерного массива по первому измерению?