## Массивы в С#

До настоящего момента мы использовали в программах простые переменные. При этом каждой области памяти, выделенной для хранения одной величины, соответствует свое имя. Если переменных много, программа, предназначенная для их обработки, получается длинной и однообразной. Поэтому в любом процедурном языке есть понятие *массива* — ограниченной совокупности однотипных величин.

Элементы массива имеют одно и то же имя, а различаются порядковым номером *(индексом).* Это позволяет компактно записывать множество операций с помощью циклов.

Массив относится к ссылочным типам данных, то есть располагается в динамической области памяти, поэтому *создание массива* начинается с выделения памяти под его элементы. Элементами массива могут быть величины как значимых, так и ссылочных типов (в том числе массивы). Массив значимых типов хранит значения, массив ссылочных типов — ссылки на элементы. Всем элементам *массива* при создании массива присваиваются значения по умолчанию: нули для значимых типов и null — для ссылочных.

Пример создания массива из 10 целых элементов:

int[] mas=new int[10];

Операция new выделяет память под 10 целых элементов, и они заполняются нулями.

Количество задается при выделении памяти и не может быть изменено впоследствии. Размерность может задаваться не только константой, но и выражением. Результат вычисления этого выражения должен быть неотрицательным целым числом. *Элементы массива нумеруются с нуля,* поэтому максимальный номер элемента всегда на единицу меньше размерности (например, в описанном выше массиве элементы имеют индексы от 0 до 9). Для *обращения к элементу массива* после имени массива указывается номер элемента в квадратных скобках, например:

mas[4], mas[i]

С элементом массива можно делать все, что допустимо для переменных того же типа. При работе с массивом автоматически выполняется контроль выхода за его границы.

#### Динамическая память

Все переменные, объявленные в программе размещаются в одной непрерывной области памяти, которую называют сегментом данных (64 Кб). Такие переменные не меняют своего размера в ходе выполнения программы и называются статическими. Размера сегмента данных может быть недостаточно для размещения больших массивов информации. Выходом из этой ситуации является использование динамической памяти. **Динамическая память** – это память, выделяемая программе для ее работы за вычетом сегмента данных, стека, в котором размещаются локальные переменные подпрограмм и собственно тела программы.

Для создания динамических переменных используют операцию new:

int [] Arr=new int[10];

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  |  | Динамическая память (куча, heap) | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
| Arr | | стек | | | | | | | | |

Массивы одного типа можно присваивать друг другу. При этом происходит присваивание ссылок, а не элементов, как и для любого другого объекта ссылочного типа, например:

i n t [ ] а = new int[10];

i n t [ ] b = а; // b и а указывают на один и тот же массив

*Одномерные массивы* используются в программах чаще всего. Варианты описания массива:

тип[] имя;

тип[] имя = new тип [ размерность ];

тип[] имя = { список\_инициализаторов };

тип[] имя = new тип [] { список\_инициализаторов };

тип[] имя = new тип [ размерность ] { список\_инициализаторов };

Примеры описаний (один пример для каждого варианта описания):

i n t [ ] а;//элементов нет

i n t [ ] b= new int[4];//элементы равны 0

i n t [ ] с = { 61. *2,* 5, -9 } ;//new не указана явно

i n t [ ] d = new i n t [ ] { 61. 2, 5. -9};//размерность вычисляется

i n t [ ] e = new int[4] { 61. *2,* 5. -9};//избыточная информация

В первом примере описана только ссылка на массив, а память под элементы массива не выделена. Если список инициализации не задан, размерность может быть не только константой, но и выражением типа, приводимого к целому. В третьем примере количество элементов вычисляется по количеству инициализирующих значений. Для полей объектов и локальных переменных можно опускать операцию new, она будет выполнена по умолчанию (оператор 2 ). Если присутствует и размерность, и список инициализаторов, размерность должна быть константой (оператор 4 ) .

**Оператор foreach** применяется для перебора элементов в специальным образом организованной группе данных. Массив является именно такой группой. Удобство этого вида цикла заключается в том, что нам не требуется определять количество элементов в группе и выполнять их перебор по индексу: мы просто указываем на необходимость перебрать все элементы группы. Синтаксис оператора:

**foreach ( тип имя in выражение ) тело цикла**

Пример

i n t [ ] а = {24, 50, 18, 3, 16, -7, 9, -1 } ;

Вывод этого массива на экран с помощью оператора foreach выглядит следующим образом:

foreach ( int х in а ) Console.WriteLine( х );

Этот оператор выполняется так: на каждом проходе цикла очередной элемент массива присваивается переменной х и с ней производятся действия, записанные в теле цикла.

При создании массива, состоящего из элементов ссылочного типа, память выделяется только под ссылки на элементы, а сами элементы необходимо разместить явным образом.

### Обработка одномерных массивов

При работе с массивами очень часто требуется одинаково обрабатывать все элементы или часть элементов массива.

Перебор элементов массива характеризуются :

* направлением перебора;
* количеством одновременно обрабатываемых элементов;
* характером изменения индексов.

По направлению перебора массивы обрабатывают:

* слева направо;
* справа налево;
* от обоих концов к середине.

Индексы могут меняться

* линейно (с постоянным шагом);
* нелинейно (с переменным шагом).

### Перебор элементов массива по одному элементу

1. двигаясь от начала массива к его концу (или в обратном направлении) с шагом 1:  
   for(int i=0;i<size;i++) <обработка a[i]>

for(int i=size-1;i>=0;i--) <обработка a[i]>

1. двигаясь от начала массива к его концу (или в обратном направлении) с шагом отличным от 1:  
   for(int i=0;i<size;i+=step) <обработка a[i]>

for(int i=size-1;i>=0;i-=step) <обработка a[i]>

1. Элементы массива можно обрабатывать по два элемента, двигаясь с обеих сторон массива к его середине:  
   int l=0,r=n-1;  
   while (l<r){  
   <обработка a[l] и a[r]>;  
   i++;j--;}

### Формирование массива с помощью датчика случайных чисел.

Для того чтобы сформировать массив необходимо организовать цикл для перебора элементов массива.

Чтобы сгенерировать последовательность псевдослучайных чисел, используется класс Random.

Алгоритм создания псевдодинамического числа.

1. Создать переменную типа Random.

Random rand=new Random();/\*для вычисления начального числа последовательности случайных чисел используется системное время\*/

1. Сформировать число с помощью одного из следующих способов:

mas\_int[i] = rand.Next();/\* целое число из диапазона 0-int32.MaxVaiue-1, включительно

mas\_int[i] = rand.Next(left, right); /\* целое число из диапазона от left до right -1 включительно

mas\_int[i] = rand.Next(right); /\* целое число из диапазона от 0 до right -1 включительно

mas\_double[i] = rand.NextDouble();/\*число (представленное в форме с плавающей точкой), которое будет больше или равно числу 0,0 и меньше 1,0

Console.WriteLine("Введите количество элементов в массиве");

int size = Convert.ToInt32(Console.ReadLine());

if (size <= 0){

Console.WriteLine ("Не правильно задан размер массива");

return;

}

Random rnd=new Random();

int[] arr = new int[size];

//вариант 1

for(int i=0;i<size;i++)

{

arr[i] = rnd.Next(1, 100);

Console.Write(arr[i].ToString("d") + " ");

}

Console.WriteLine();

## //вариант 2

for (int i = 0; i < size; i++)

arr[i] = rnd.Next(1, 100);

foreach (int x in arr)

Console.Write(x.ToString("d") + " ");

Console.WriteLine();

**Задача.** Найти сумму элементов массива с четными индексами.

Для решения этой задачи можно предложить несколько способов.

1. Массив перебирается с шагом 2 и все элементы суммируются.
2. Массив перебирается с шагом 1 и суммируются только элементы, имеющие четные индексы. Для проверки на четность используется операция получения остатка от деления на 2.

//1 способ

int summa = 0;

int tek = 1;

for (; tek < size; tek += 2) summa += arr[tek];

Console.WriteLine("Сумма элементов массива с четными номерами=" + summa);

//2 способ

int summa = 0;

int tek = 1;

for (; tek < size; tek ++)

if(i%2==0)

summa += arr[tek];

Console.WriteLine("Сумма элементов массива с четными номерами=" + summa);

### Перебор элементов массива по два элемента

1. Элементы массива можно обрабатывать по два элемента, двигаясь от начала к концу, с шагом 1(т. е. обрабатываются пары элементов a[0]и a[1], a[1]и a[2] и т. д.)  
   for(i=0;i<size-1;i++)  
   <обработка a[i] и a[i+1]>
2. Элементы массива можно обрабатывать по два элемента, двигаясь от начала к концу с шагом 2(т. е. обрабатываются пары элементов a[0]и a[1], a[2]и a[3] и т. д.)  
   i=1;  
   while(i<size){  
   <обработка a[i] и a[i+1]>  
   i:=i+2;}

### Классы задач по обработке массивов

1. К задачам 1 класса относятся задачи, в которых выполняется однотипная обработка всех или указанных элементов массива. Решение таких задач сводится к установлению того, как обрабатывается каждый элемент массива или указанные элементы, затем подбирается подходящая схема перебора, в которую вставляются операторы обработки элементов массива. Примером такой задачи является нахождение среднего арифметического элементов массива.
2. К задачам 2 класса относятся задачи, в которых изменяется порядок следования элементов массива.
3. К задачам 3 класса относятся задачи, в которых выполняется обработка нескольких массивов или подмассивов одного массива. Массивы могут обрабатываться по одной схеме – синхронная обработка или по разным схемам – асинхронная обработка массивов.
4. К задачам 4 класса относятся задачи, в которых требуется отыскать первый элемент массива, совпадающий с заданным значением – поисковые задачи в массиве.

### Задачи 1 класса

Решение таких задач сводится к установлению того, как обрабатывается каждый элемент массива или указанные элементы, а затем подбирается подходящая схема перебора, в которую подставляются операторы обработки.

Примеры задач 1 класса:

1. Дан массив целых чисел. Найти количество четных элементов массива.
2. Дан массив целых чисел. Найти среднее арифметическое элементов массива.
3. Дан массив целых чисел. Найти максимальный (минимальный) элемент массива.
4. Дан массив целых чисел. Найти номер максимального (минимального) элемента массива.

### Задачи 2 класса

Для изменения порядка следования используют:

1. Обмен элементов массива.
2. Сдвиг одного элемента на место другого элемента.

Обмен элементов внутри массива выполняется с использованием вспомогательной переменной:

//обмен a[i]и a[j]элементов массива.

temp=a[i];a[i]=a[j]; a[j]=temp;

Cдвиг одного элемента на место другого элемента:

a[i]=a[i+1];//сдвиг влево

a[i]=a[i-1];// сдвиг вправо

Примеры задач 2 класса:

1. Дан массив целых чисел. Перевернуть массив.

//переворот

int left = 0, right = size - 1, temp;

while (left < right)

{

temp = arr[left];

arr[left] = arr[right];

arr[right] = temp;

left++; right--;

}

foreach (int x in arr) Console.Write(x+ " ");

Console.WriteLine();

1. Дан массив целых чисел. Поменять местами пары элементов ( 1-ый и 2-ой, 3-ий и 4-ый, и т.д.)

//меняем пары элементов

int tek = 0;

for (; tek < size - 1; tek += 2)

{

temp = arr[tek];

arr[tek] = arr[tek + 1];

arr[tek + 1] = temp;

}

foreach (int x in arr) Console.Write(x + " ");

Console.WriteLine();

1. Сдвинуть элементы массива на k элементов влево (вправо).

int numberOfCycles;

do

{

Console.WriteLine("Введите на вколько элементов нужно сдвинуть массив");

numberOfCycles=Convert.ToInt32(Console.ReadLine());

if(numberOfCycles<0) Console.WriteLine("Ошибка! число должно быть больше 0");

if(numberOfCycles==0)Console.WriteLine("Вы уверены? С массивом ничего не поизойдет!");

}while(numberOfCycles<=0);

int j;

//сдвиг влево

for (int count = 0; count < numberOfCycles; count++)

{

temp = arr[0];

for (j = 0; j < size - 1; j++) arr[j] = arr[j + 1];

arr[size - 1] = temp;

}

foreach (int x in arr) Console.Write(x + " ");

Console.WriteLine();

//сдвиг вправо

for (int count = 0; count < numberOfCycles; count++)

{

temp = arr[size-1];

for (j = size - 1; j >0; j--) arr[j] = arr[j - 1];

arr[0] = temp;

}

### Задачи 3-ого класса

При синхронной обработке массивов индексы при переборе массивов меняются одинаково.

Примеры задач 3 класса:

1. Заданы два массива из n целых элементов. Получить массив c, где c[i]=a[i]+b[i].

for(int i=0;i<n;i++) c[i]=a[i]+b[i];

При асинхронной обработке массивов индекс каждого массива меняется по своей схеме.

1. В массиве целых чисел все отрицательные элементы перенести в начало массива.

int tek\_temp=0;

int []arr\_temp=new int[size];

for(tek=0 ;tek<size;tek++)

if (arr[tek] < 0)

{

arr\_temp[tek\_temp] = arr[tek];

tek\_temp++;

}

for (tek = 0; tek < size; tek++)

if (arr[tek] >= 0)

{

arr\_temp[tek\_temp] = arr[tek];

tek\_temp++;

}

foreach (int x in arr\_temp) Console.Write(x + " ");

Console.WriteLine();

### Задачи 4-ого класса

В поисковых задачах требуется найти элемент, удовлетворяющий заданному условию. Для этого требуется организовать перебор массива и проверку условия. Но при этом существует две возможности выхода из цикла:

1. нужный элемент найден ;
2. элемент не найден, но просмотр массива закончен.

Пример задачи 4 класса:

Найти первое вхождение элемента К в массив целых чисел.

//способ 1

int numberForFind;

int index=-1;

Console.WriteLine("Введите число для поиска");

numberForFind=Convert.ToInt32(Console.ReadLine());

for (tek = 0; tek < size; tek++ )

if (arr[tek] == numberForFind)

{

index = tek;

break;

}

if (index<0) Console.WriteLine("Элемент не найден");

else Console.WriteLine("Номер Элемента {0} равен {1}", numberForFind, index+1);

//способ 2

int numberForFind;

int index=-1;

Console.WriteLine("Введите число для поиска");

numberForFind=Convert.ToInt32(Console.ReadLine());

for (tek = 0; tek < size && arr[tek] != numberForFind; tek++) ;

if (tek == size) Console.WriteLine("Элемент не найден");

else Console.WriteLine("Номер Элемента {0} равен {1}", numberForFind, tek+1);

.

### Сортировка массивов

Сортировка – это процесс перегруппировки заданного множества объектов в некотором установленном порядке.

#### Сортировка с помощью включения

Элементы массива делятся на уже готовую последовательность и исходную. При каждом шаге, начиная с i=2, из исходной последовательности извлекается i-ый элемент и вставляется на нужное место готовой последовательности, затем i увеличивается на 1 и т. д.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 44 | 55 | 12 | 42 | 94 | 18 |
| готовая | исходная | | | | |

В процессе поиска нужного места осуществляются пересылки элементов больше выбранного на одну позицию вправо, т. е. выбранный элемент сравнивают с очередным элементом отсортированной части, начиная с j=i-1. Если выбранный элемент больше a[i], то его включают в отсортированную часть, в противном случае a[j] сдвигают на одну позицию, а выбранный элемент сравнивают со следующим элементом отсортированной последовательности. Процесс поиска подходящего места заканчивается при двух различных условиях:

* если найден элемент a[j]>a[i];
* достигнут левый конец готовой последовательности.

int j,el;

for (i = 1; i < size; i++)

{

el = arr[i];//запомнили элемент, который будем вставлять

j = i - 1;

while ( j >= 0&&el < arr[j] )//поиск подходящего места

{

arr[j + 1] = arr[j];//сдвиг вправо

j--;

}

arr[j + 1] = el;//вставка элемента

}

foreach ( int x in arr) Console.Write(x + " ");

Console.WriteLine();

#### Сортировка методом простого выбора

Выбирается минимальный элемент массива и меняется местами с первым элементом массива. Затем процесс повторяется с оставшимися элементами и т. д.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 44 | 55 | 12 | 42 | 94 | 18 |
| 1 | 2 | 3  Мин | 4 | 5 | 6 |

int min, n\_min, j;

for (i = 0; i < size - 1; i++)

{

min = arr[i]; n\_min = i;//поиск минимального

for (j = i + 1; j < size; j++)

if (arr[j] < min)

{

min = arr[j];

n\_min = j;

}

arr[n\_min] = arr[i];//обмен

arr[i] = min;

}

foreach ( int x in arr) Console.Write(x + " ");

Console.WriteLine();

#### 19.5.3. Сортировка методом простого обмена

Сравниваются и меняются местами пары элементов, начиная с последнего. В результате самый маленький элемент массива оказывается самым левым элементом массива. Процесс повторяется с оставшимися элементами массива.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 44 | | | 55 | | | | 12 | | 42 | 94 | 18 |
| 1 | | | 2 | | | | 3 | | 4 | 5 | 6 |
|  |  |  | |  |  |  | |

int j;

for(i=1;i<size;i++)

for(j=size-1;j>=i;j--)

if(arr[j]<arr[j-1])

{

int temp=arr[j];

arr[j]=arr[j-1];

arr[j-1]=temp;

}

foreach (int x in arr) Console.Write(x + " ");

Console.WriteLine();

#### Поиск в отсортированном массиве

В отсортированном массиве используется дихотомический (бинарный) поиск. При последовательном поиске требуется в среднем n/2 сравнений, где n – количество элементов в массиве. При дихотомическом поиске требуется не более m сравнений, если n- m-ая степень 2, если n не является степенью 2, то n<k=2m.

Массив делится пополам S:=(L+R)/ 2+1 и определяется в какой части массива находится нужный элемент Х. Так как массив упорядочен, то, если a[S]<X – искомый элемент находится в правой части массива, иначе – находится в левой части. Выбранную часть массива снова надо разделить пополам и т. д., до тех пор, пока границы отрезка L и R не станут равны.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 8 | 10 | 11 | 15 | 19 | 21 | 23 | 37 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

L S R

Console.WriteLine("Введите число для поиска");

int numberForFind=Convert.ToInt32(Console.ReadLine());

int left = 0, right = size - 1, sred;

do

{

sred = (left + right) / 2;//средний элемент

if (arr[sred] < numberForFind) left = sred + 1;//перенести левую границу

else right = sred;//перенести правую границу

} while (left != right);

if (arr[left]==numberForFind) Console.WriteLine("Номер Элемента {0} равен {1}", numberForFind, left+1);

else Console.WriteLine("Элемент не найден");