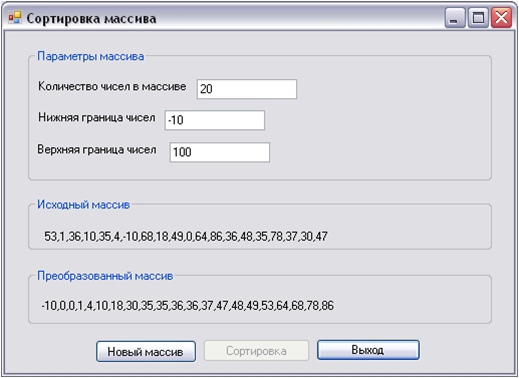
**Пример выполнения:**

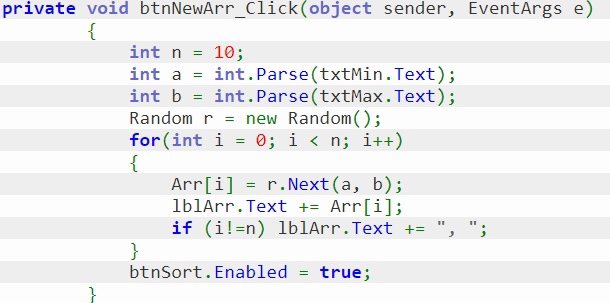


Форма «Сортировка массива» (Рисунок 1)

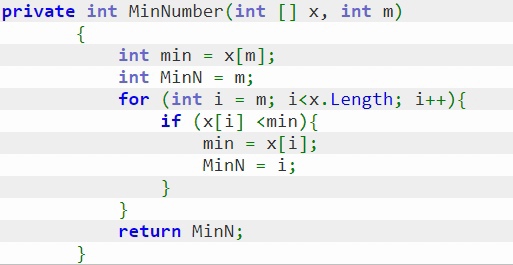
1. Рассмотрим задачу упорядочивания массива в простейшей постановке: дан числовой массив *х1*, …, *хn*, элементы которого попарно различны, требуется переставить элементы массива так, чтобы после перестановки они были упорядочены в порядке возрастания: *х1*< … < *хn*. Опишем алгоритм решения этой задачи.
2. Используем алгоритм сортировки выбором. Суть его такова: очевидно, что первое место в массиве должен занять наименьший элемент, второе – наименьший из всех остальных элементов и т.д. Пусть *x1*, …, *xi-1* уже получили нужные значения. Тогда определение индекса *k* наименьшего из *xi*, *xi+1*, … *xn* и перестановка *xi* c *xk* приведут к тому, что *x1*, …, *xi* будут иметь нужные значения. Составим программу для этого алгоритма.
3. Создайте новый проект и разместите элементы управления (*txtN, txtMin, txtMax, lblArr, lblResult, btnNewArr, btnSort* и *btnExit*) на форме так, как показано на рисунке (*рисунок 1*). Используйте элемент *GroupBox* для группировки.
4. Установите значение для свойства *Enabled*кнопки *btnSort* равным *False* (т.о., заблокировав кнопку).
5. Самостоятельно запрограммируйте кнопку *Выход*. (***Это же вы сможете, только не говорите, что нет***)
6. Заблокируйте текстовое поле *txtN*, т.к. мы будем использовать 10 элементов массива (свойство Enabled = false).
7. В разделе описаний объявите массив из 10 элементов следующим образом:

Практическая работа №8, изображение №3

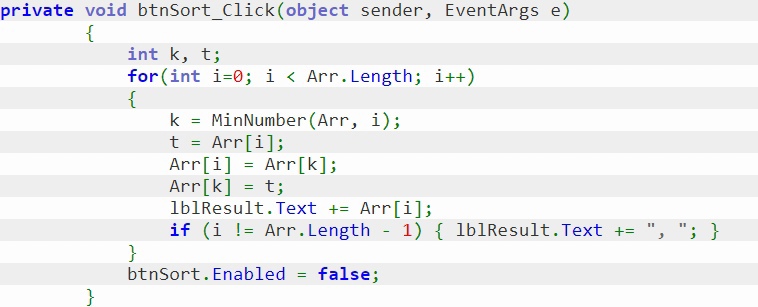
1. Теперь необходимо инициализировать наш массив целыми числами из некоторого промежутка. Процедура обработки события *Click* для кнопки *btnNewArr* выглядит следующим образом:



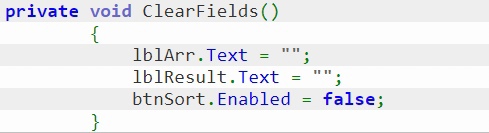
1. Наконец, реализуем описанный выше алгоритм сортировки в процедуре обработки события *Click* для кнопки *btnSort*. Предварительно опишем функцию *MinNumber*, которая осуществляет поиск позиции минимального элемента массива:



1. Теперь опишем главный фрагмент кода, который вызывает функцию поиска позиции минимума, начиная с нулевой позиции массива; затем, после выполнения функции, на нулевое место становится минимальный элемент, а функция вызывается с аргументом уже первой (а не нулевой) позиции (затем второй позиции и т.д.):



1. Запустите и отладьте (***догадаетесь, что от вас требуют***) приложение. Вы, очевидно, заметили, что программа начинает некорректно работать при повторном нажатии на кнопку *Новый массив*. Для того чтобы выправить эти дефекты, создадим процедуру для очистки окон *ClearFiealds*:



1. Вызовем созданную подпрограмму в начале процедуры обработки события *Click* для кнопки *btnNewArr*, а также в процедурах обработки события *Change* для текстовых окон *txtMin* и *txtMax* следующим образом:
2. Запустите и отладьте приложение.
3. Самостоятельно разработать программу для реализации *алгоритма сортировки вставками*. Суть этого алгоритма в следующем: просматриваются последовательно элементы массива *х2*, …,*хn* и каждый новый элемент *xi* вставляется на подходящее место в уже упорядоченную совокупность *x1*, …, *xi-1*. Это место определяется последовательным сравнением *xi*с упорядоченными элементами *x1*, …, *xi-1*. При решении этой задачи необходимо описать и использовать функцию *Insertion*, которая вставляет *xi*на подходящее место в уже упорядоченную последовательность *x1*, …, *xi-1*.