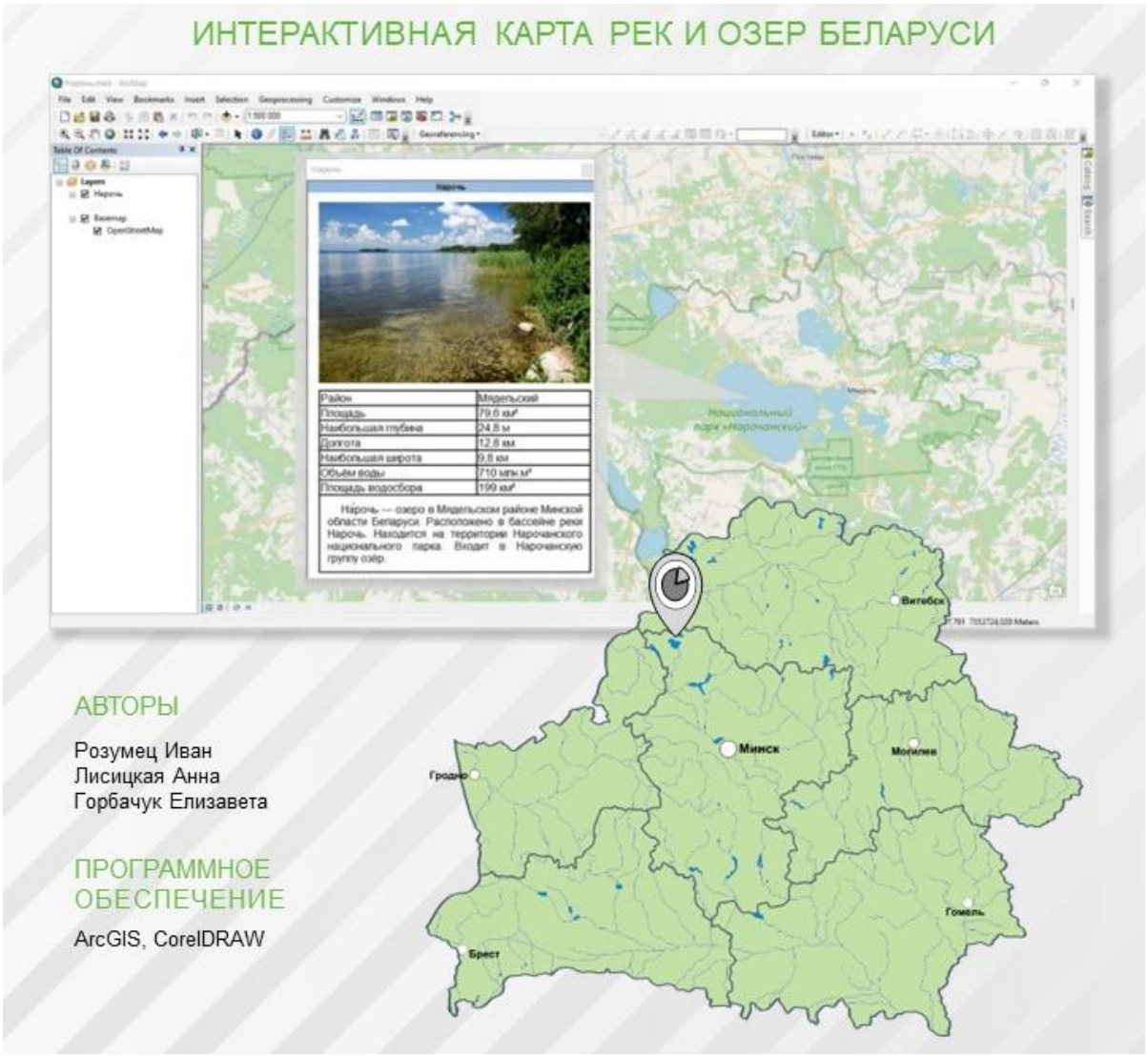


Рисунок 3 – Пример интерактивной карты рек и озер Республики Беларусь

Список цитируемых источников

1. Научная электронная библиотека «КиберЛенинка» [Электронный ресурс] / Картография. – Режим доступа: https://cyberleninka.ru/article/n/opisanie-kontseptsii-interaktivnoy-karty-1/viewer. – Дата доступа: 01.04.2022.

2. Базы данных. Учебное пособие [Электронный ресурс] / Информатика и вычислительная техника. – Режим доступа: https://siblec.ru/informatika-i-vychislitelnaya-tekhnika/bazy-dannykh. – Дата доступа: 02.04.2022.

УДК 681.5

Евсеенко К. А., Микуц А. Н.
Научный руководитель: ст. преподаватель Клопоцкий А. А.

РАЗРАБОТКА УПРАВЛЯЮЩИХ ПРОГРАММ С ИНТЕГРИРОВАНИЕМ PYTHON В DELPHI

Python для Delphi (P4D) – это набор бесплатных компонентов, которые превращают dllPython в Delphi и Lazarus (FPC). Они позволяют легко выполнять сценарии Python, создавать новые модули Python и новые типы Python. Вы мо-

жете создавать расширения Python в виде dll и многое другое. P4D обеспечивает различные уровни функциональности:

- Низкоуровневый доступ к API Python.
- Высокоуровневое двунаправленное взаимодействие с Python.
- Доступ к объектам Python с использованием пользовательских вариантов Delphi (VarPyth.pas)
- Обертывание объектов Delphi для использования в сценариях Python с использованием RTTI (WrapDelphi.pas).

P4D позволяет очень легко использовать Python в качестве языка сценариев для приложений Delphi. Он поставляется с обширным набором демонстраций и учебных пособий.

Зачем разработчику Delphi добавлять Python в свой набор инструментов? Все дело в доступе к библиотеке и возможности сценариев. Библиотека Python4Delphi (P4D) с открытым исходным кодом, разработанная Кириакосом Влахосом, автором популярной интегрированной среды разработки PythonPyScripter, позволяет вам как разработчику Delphi использовать всю коллекцию библиотек Python непосредственно из Delphi. Это также упрощает выполнение скриптов Python, создание новых модулей Python и новых типов Python непосредственно из вашего приложения Delphi.

Python4Delphi решает проблему загрузки основной библиотеки DLL Python в программу Delphi, встраивая интерпретатор Python в ваше приложение Delphi, но также имеет несколько демонстраций для обратного – для написания расширения с использованием Delphi.

Чтобы иметь представление, как языки программирования взаимодействуют друг с другом, рассмотрим следующий пример – сортировку методом пузырька.

В чем суть? При работе с массивами данных не редко возникает задача их сортировки по возрастанию или убыванию, т. е. упорядочивания. Это значит, что элементы того же массива нужно расположить строго по порядку. Например, в случае сортировки по возрастанию предшествующий элемент должен быть меньше последующего (или равен ему).

Что делать? Существует множество методов сортировки. Одни из них являются более эффективными, другие – проще для понимания. Достаточно простой для понимания является сортировка методом пузырька, который также называют методом простого обмена. В чем же он заключается и почему у него такое странное название "метод пузырька"?

А почему же такое название? Как известно воздух легче воды, поэтому пузырьки воздуха всплывают. Это просто аналогия. В сортировке методом пузырька по возрастанию более легкие (с меньшим значением) элементы постепенно "всплывают" в начало массива, а более тяжелые друг за другом опускаются на дно (в конец массива).

А зачем же их использовать в тандеме, если проще написать сразу в Python, не применяя Delphi? Достоинства следующие:

- Визуализация в Delphi [1], которой нет в Python [2].
- Высокое быстродействие в Delphi; загрузка и обработка кода в Python оставляет желать лучшего.
- Высокое качество визуализации.

С чего же начать? Взаимодействие Python и Delphi выглядит так: весь основной код содержит Python, а в Delphi лишь – графическая оболочка. Следовательно, сортировку будем писать в Python.

Что понадобится? Python славится не только простой кода, но и огромным множеством библиотек, одна из которых и понадобится в нашем примере – библиотека сортировки:

```
from SortModule import getvalue, swap
```

В нашем примере мы будем сортировать три модуля, следовательно, нам нужно создать и описать три функции. Опишем первую функцию:

```
def SortFunc1(handle, low, high):
    for i in range(low, high):
        for j in range(low + 1, high):
            if getvalue(handle, j - 1) > getvalue(handle, j):
                swap(handle, j - 1, j)
```

Что мы видим? Создается цикл, в котором описывается скорость, плавность и метод сортировки. В нашем случае сортировка будет самой медленной из всех трёх примеров, но при этом и самой плавной.

Опишем вторую функцию:

```
def SortFunc2(handle, low, high):
    for i in range(low, high - 1):
        for j in range(i + 1, high):
            if getvalue(handle, i) > getvalue(handle, j):
                swap(handle, i, j)
```

Как видно, глобальных изменений здесь не последовало, лишь небольшое увеличение скорости сортировки.

Опишем третью функцию. Третья функция будет самой быстрой из всех, которые представлены в нашем примере, вплоть до того, что не вооруженным глазом ее не заметишь. При демонстрации сразу трех функций это будет заметно.

Работа в Python закончена, остаётся лишь сохранить наш код в отдельном файле и загрузить его в Delphi, когда это будет необходимо.

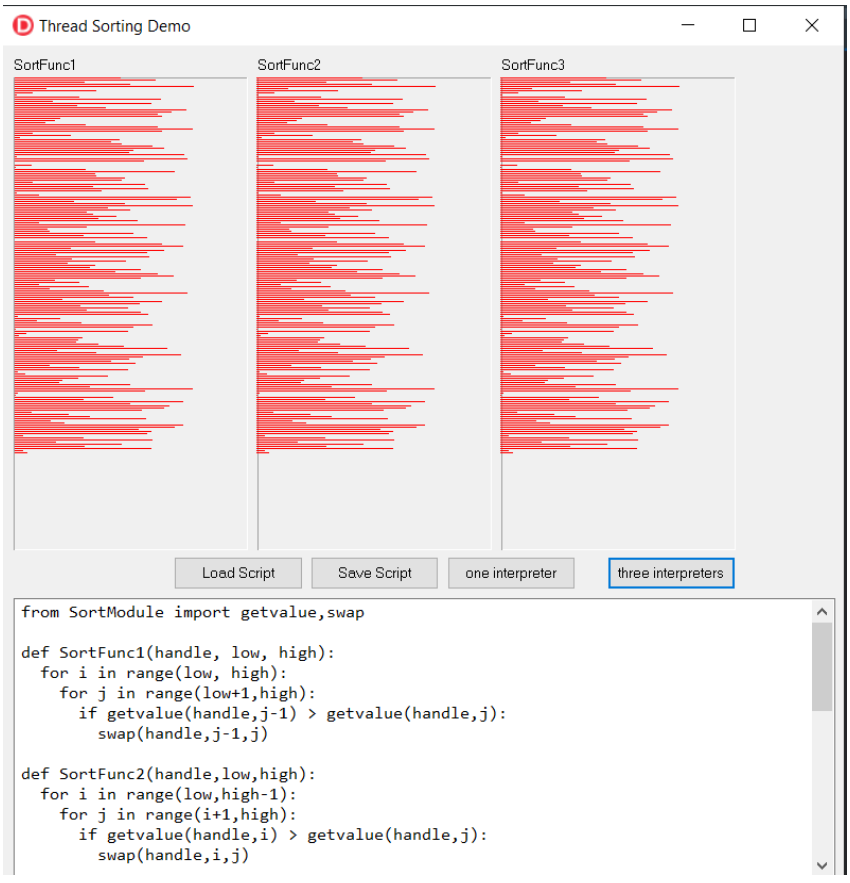
Переходим в Delphi. Delphi славится своей графической визуализацией, поэтому метод сортировки запустится как нельзя кстати. Переходим в Design. С помощью библиотеки компонентов добавим те, которые необходимы для качественной визуализации. Добавим три компонента SortBox – это те области, в которых будет происходить сортировка. 4 компонента Button – простыми словами «кнопка». Кнопка загрузки скрипта Python, его сохранения, и две кнопки для сортировки: для сортировки 1 функции и трех функций сразу.

```
def SortFunc3(handle, low, high):
    Lo = low
    Hi = high - 1
    Mid = getvalue(handle, (Lo + Hi) // 2)
    while 1:
        while getvalue(handle, Lo) < Mid:
            Lo = Lo + 1
        while getvalue(handle, Hi) > Mid:
            Hi = Hi - 1
        if Lo <= Hi:
            swap(handle, Lo, Hi)
            Lo = Lo + 1
            Hi = Hi - 1
        if Lo > Hi:
            break
    if Hi > low:
        SortFunc3(handle, low, Hi + 1)
    if Lo < high - 1:
        SortFunc3(handle, Lo, high)
```

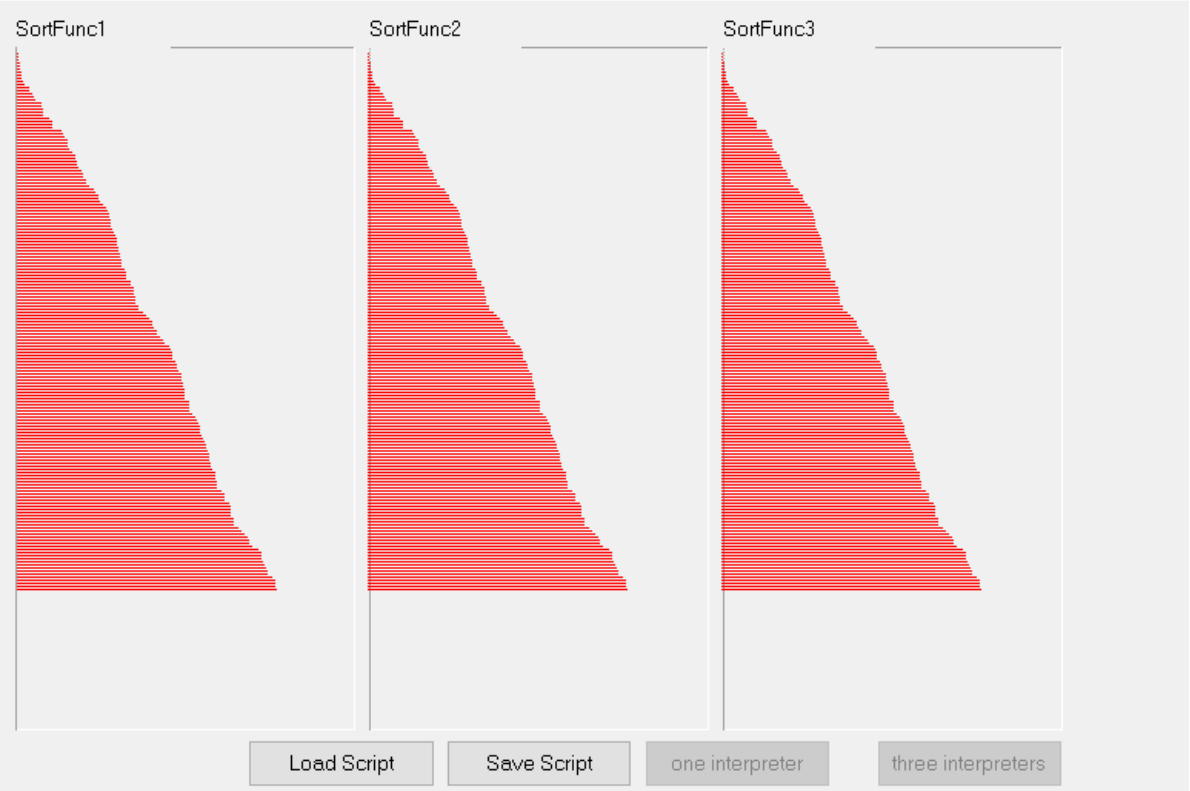
Осталось пару компонентов, которые мы возьмем из библиотеки P4D:

- PythonEngine – загружает и делает доступным Python. Доступ (low-level) к Python API.
- PythonModule – создает модуль Python в Delphiи обеспечивает доступ к нему из Python.
- SortModule – собственно, модуль сортировки.
- PythonMemo – область, где будет показан загруженный нами код Python.

Таким образом, собрав и разместив все компоненты по своим местам и описав все компоненты в Delphi, загрузим код из Python с помощью кнопки Load Script и увидим разбросанные пиксели, сортировка еще не запущена.



Выбрав кнопку one interpreter или three interpreters, запустим данный скрипт и увидим сортировку (картинками, к сожалению, не передать момент самой сортировки).



Результатом является расположение значений по возрастанию. Таким образом, совместное использование двух программных пакетов облегчает процесс программирования при расширении возможностей визуализации.

Список цитированных источников

1. Архангельский, А. Я. Программирование в Delphi 7 / А. Я. Архангельский. – М. : ООО «Бином-Пресс», 2003. – 1152 с.

2. Саммерфилд, М. Программирование на Python 3. Подробное руководство: пер. с англ. / М. Саммерфилд. – СПб. : Символ-Плюс, 2009. – 608 с.

УДК 519.6

Римашевская А. И.
Научный руководитель: ассистент Сидак С. В.;
ст. преподаватель Рамская Л. К.

ФРАКТАЛЬНЫЙ АНАЛИЗ ФИНАНСОВЫХ ВРЕМЕННЫХ РЯДОВ

Развитие современных компьютерных технологий, совершенствование методов математического моделирования, автоматизация бизнес-процессов позволяют использовать более сложные математические методы нахождения параметров, отражающих скрытые, неявные свойства финансовых процессов. Одним из таких инструментов, получившим в настоящее время популярность в различных областях науки, является фрактальный анализ [1, с. 2]. Суть данного метода заключается в том, что состояние системы, в котором она находится в настоящий момент, формируется на основе предыдущих состояний или процессов.