DOI: 10.32517/2221-1993-2024-23-4-66-75

А. Д. Алексеевец
Средняя школа № 71, г. Ярославль, Россия

Н. И. Заводчикова
Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия;
Средняя школа № 76, г. Ярославль, Россия

СОЗДАНИЕ ОКОННЫХ ПРИЛОЖЕНИЙ В PYTHON СРЕДСТВАМИ БИБЛИОТЕКИ PYQT5

Аннотация

В содержание раздела «Программирование» на углубленном уровне изучения курса информатики в старшей школе входит рассмотрение вопросов, связанных с проектированием интерфейса пользователя. Если изучение основ программирования происходит с помощью языка Python, то целесообразно и разработку оконных приложений осуществлять с использованием этого языка. В последнее время все большую популярность приобретает библиотека PyQt, что делает актуальным создание методических материалов для знакомства школьников с возможностями указанной библиотеки.

Обучение целесообразно осуществлять в соответствии с технологией реверс-инжиниринга, состоящей в том, что учащиеся не пишут код программы с нуля, а исследуют готовый продукт. На первом этапе можно организовать деятельность школьников с помощью выполнения заданий на исследование готовой программы; на следующем шаге предложить проанализировать внешний вид готового приложения, выделить его элементы и подобрать инструменты для их создания; в заключение предоставить дополнительные задачи, предполагающие самостоятельное изучение учащимися возможностей класса, не рассмотренного ранее.

В статье предложены справочный материал для знакомства школьников с основными возможностями библиотеки, задания для лабораторных и домашних работ.

Ключевые слова: обучение программированию, разработка оконных приложений, реверс-инжиниринг, набор задач.

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_04-2024/

Контактная информация

Алексеевец Анастасия Дмитриевна, учитель информатики, средняя школа № 71, г. Ярославль, Россия; *адрес:* 150040, Россия, г. Ярославль, ул. Свердлова, д. 62; *e-mail:* aadyar@yandex.ru

Заводчикова Надежда Ивановна, канд. пед. наук, доцент,
доцент кафедры теории и методики обучения информатике, Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия; *адрес:* 150000, Россия, г. Ярославль, ул. Республиканская, д. 108/1;
учитель информатики, средняя школа № 76, г. Ярославль, Россия; *адрес:* 150031, Россия, г. Ярославль, ул. Автозаводская, д. 69д; *e-mail:* zaw-nadejda@yandex.ru

A. D. Alekseevets
School 71, Yaroslavl, Russia
N. I. Zavodchikova
Yaroslavl State Pedagogical University named after K. D. Ushinsky, Yaroslavl, Russia;
School 76, Yaroslavl, Russia

CREATING WINDOW APPLICATIONS IN PYTHON USING THE PYQT5 LIBRARY TOOLS

Abstract

The section "Programming" at the advanced level of studying the informatics course in high school includes consideration of issues related to the design of the user interface. If the study of the basics of programming is carried out using the Python language, it is reasonable to develop window applications using this language. Recently, the PyQt library has become increasingly popular, which makes it relevant to create methodological materials to familiarize schoolchildren with the capabilities of this library.

It is advisable to carry out training in accordance with the reverse engineering technology, which consists in the fact that students do not write program code from scratch, but explore the finished product. At the first stage, you can organize the activities of schoolchildren by completing tasks to study the finished program; at the next step, offer to analyze the appearance of the finished application, highlight its elements and select tools for their creation; finally, provide additional tasks that involve independent study by students of the class features not considered earlier.

The article offers reference material for familiarizing schoolchildren with the main capabilities of the library, tasks for lab sessions and home assignment.

Keywords: training in programming, development of window applications, reverse engineering, set of tasks.

1. Введение

Раздел «Алгоритмы и программирование» неизменно является одним из основных в школьном курсе информатики. За последние 40 лет накоплен огромный опыт преподавания этой темы на разных этапах обучения. Младшие школьники программируют, как правило, с использованием интерактивных сред с виртуальными управляемыми объектами. В основной школе в качестве первого языка для изучения программирования в последнее время выбирают язык Python [4, 7, 9]. В средней школе в содержание раздела «Программирование» на углубленном уровне входит рассмотрение вопросов, связанных с использованием готовых управляемых элементов для построения интерфейса [12]. В федеральной рабочей программе указано на необходимость проведения практической работы «Разработка программы с графическим интерфейсом» [12].

При изучении в качестве первого языка программирования Pascal наиболее приемлемым являлся переход от Pascal к Delphi [5, 6]. Рассмотрение основ программирования на языке Python вынуждает и разработку оконных приложений реализовывать на этом языке.

Программы с графическим интерфейсом на языке Python обычно строятся на основе графических библиотек: *tkinter*, *wxPython*, *PyGTK*, *PyQt*. В учебнике К. Ю. Полякова, Е. А. Еремина [8] описано использование библиотеки *tkinter*, не требующей установки. В последнее время все большую популярность приобретает библиотека *PyQt5*, что делает актуальным создание методических материалов для изучения школьниками возможностей данной библиотеки.

В данной статье предложен справочный материал для знакомства школьников с основными возможностями библиотеки *PyQt5*, задания для лабораторных и домашних работ.

2. Общие сведения о библиотеке PyQt

Для создания оконного приложения необходимо подключить как минимум два модуля:

- модуль *PyQt5.QtWidgets* содержит классы, реализующие компоненты графического интерфейса: окна, надписи, кнопки, текстовые поля и др. [10]. Так как *QtWidgets* содержит много необходимых для разработки приложения объектов, из него целесообразно импортировать все его объекты;
- из модуля *sys* потребуется только функция *exit*(), позволяющая завершить выполнение программы, импортировать все объекты данного модуля нет необходимости.

Создание приложения осуществляется в виде экземпляра класса *QApplication*. Следует помнить, что в программе всегда должен быть объект приложения, причем только один [10]. Объект «окно» создается как экземпляр класса *QWidget*.

В поле окна можно расположить различные компоненты. Наиболее часто в окне располагаются надписи для вывода подсказки пользователю (класс *QLabel*), кнопки для запуска обработчика событий (класс *QPushButton*) и текстовые поля для ввода информации (класс *QLineEdit*). Для создания экземпляров указанных классов необходимо указать имя объекта слева от знака «=», класс объекта и его параметры справа от знака «=» (табл. 1). В параметре *parent* указывается ссылка на родительский компонент (в предложенных ниже заданиях это будет имя окна приложения); если этот параметр не указан, то компонент будет обладать своим окном.

В таблице 2 представлен необходимый для решения предложенных ниже задач минимальный набор методов классов *QWidget*, *QLabel* и *QLineEdit*, приведен пример обработки сигнала для класса *QPushButton*.

Таблица 1

Создание основных элементов интерфейса оконного приложения

Класс	Создание экземпляра класса
Приложение	`<Объект> = QApplication([])`
Окно	`<Объект> = QWidget()`
Надпись	`<Объект> = QLabel(<parent>)`
Командная кнопка	`<Объект> = QPushButton(<текст>, <parent>)`
Однострочное текстовое поле	`<Объект> = QLineEdit(<parent>)`

Таблица 2

Справочный материал, необходимый для выполнения заданий лабораторной работы

Некоторые методы классов *PyQt5.QtWidgets*		
Класс	Метод	Описание
QWidget	`setWindowTitle(<Текст>)`	Задает текст, который будет отображен в заголовке окна
	`show()`	Отображает окно

Окончание табл. 2

Класс	Метод	Описание
	`setGeometry(<x>,<y>,<width>,<height>)`	Устанавливает положение на экране (*x*, *y*) и размеры (*width*,*height*) объекта. Метод применим ко всем визуальным объектам (наследуется всеми классами визуальных объектов)

	`x()`
	`y()`
	`width()`
	`height()`
		Возвращает абсциссу левого верхнего угла окна
		Возвращает ординату левого верхнего угла окна
		Возвращает ширину окна
		Возвращает высоту окна

	`move (<x>,<y>)`	Перемещает окно так, что его левый верхний угол занимает позицию (`<x>,<y>`)
Пример:	`window.move(window.x(),window.y()-10)`	Смещение окна с именем `window` вниз на 10 пикселей
QLabel	`setText (<Текст>)`	Задает текст, который будет отображен на надписи
	`setNum (<Число>)`	Преобразует целое или вещественное число в строку и отображает ее на надписи
QLineEdit	`text ()`	Возвращает текст, содержащийся в поле
Обработка сигналов *QPushButton*		
	`clicked.connect (<имя функции>)`	Установление связи между сигналом, который генерируется при нажатии на кнопку, и функцией, которая должна выполняться при поступлении этого сигнала
Пример:	`bt.clicked.connect(check_start)`	При нажатии кнопки с именем `bt` будет вызвана функция `check_start`

Структура PyQt-программы представлена в листинге 1.

В последней строке программы происходит запуск цикла обработки событий. После вызова метода *exec_*() для приложения *app* запускается бесконечный цикл, в котором постоянно сканируются поступающие сигналы, и, если программа «знает», как их обработать, то они обрабатываются. При работе бесконечного цикла программа может совершать действия (вызов функции, запуск метода и т. п.) только при наступлении какого-либо события, например, при нажатии пользователем кнопки. При поступлении сигнала цикл прерывается и управление передается в функцию обработки сигналов. После завершения работы обработчика управление опять передается основному циклу приложения. Цикл автоматически прерывается при закрытии окна. Инструкции, расположенные после вызова метода *exec_*(), будут выполнены только после завершения работы всего приложения.

3. Этапы организации познавательной деятельности обучающихся

Знакомство с возможностями библиотеки целесообразно осуществлять в соответствии с технологией ***реверс-инжиниринга***, состоящей в том, что учащиеся не пишут код программы с нуля, а исследуют готовый продукт.

Организация познавательной деятельности учащихся при разработке программ средствами библиотеки PyQt5 может быть следующей:

1) Обсуждение особенностей современных прикладных программ; повторение основных понятий объектно-ориентированного программирования (объект, класс, свойства и методы, точечная нотация и т. п.); знакомство с понятиями «сообщение», «событие» [8].
2) Выполнение заданий на анализ готовой программы, изменение отдельных ее параметров в соответствии с новыми требованиями.

```
from PyQt5.QtWidgets import *    # импорт модуля PyQt5.QtWidgets
import sys                       # импорт модуля sys
<описание функции обработки сигналов>
app = QApplication([])           # создание приложения с именем app
window = QWidget()               # создание окна с именем window
<создание элементов интерфейса и установка связи между сигналами и обработчиками>
window.show()                    # отображение окна с именем window
sys.exit(app.exec_())            # запуск цикла обработки событий
```

Листинг 1

3) Выполнение заданий на анализ внешнего вида готового приложения, выделение его элементов, подбор инструментов для его разработки на основе справочной информации под руководством учителя.
4) Самостоятельное решение задач на анализ интерфейса и разработку приложений с небольшим количеством объектов.
5) Разработка интерфейса и реализация приложения в соответствии с предложенными требованиями.
6) Решение дополнительных задач на самостоятельное изучение возможностей некоторого не рассмотренного ранее класса.

Этап 1 можно реализовать в форме фронтального обсуждения, используя вопросы после параграфов № 42–45 учебника К. Ю. Полякова, Е. А. Еремина [8].

Этапы 2 и 3 целесообразно осуществлять на уроке в виде самостоятельной работы учащихся, сопровождаемой консультациями учителя. В качестве справочных материалов можно предложить школьникам содержание раздела 2 настоящей статьи. Реализация указанных этапов занимает примерно два академических часа.

Этап 4 представляет собой домашнюю работу: школьники могут выполнить как одно, так и несколько из предложенных ниже заданий.

На наш взгляд, перечисленных четырех этапов вполне достаточно для соблюдения требований федеральной рабочей программы.

Если учащиеся проявили интерес к предложенной теме и резерв учебного времени позволяет, то следующие два урока можно посвятить разработке приложения «Угадай число», а в качестве домашней работы предложить школьникам задания на усовершенствование разработанного приложения.

Далее мотивированным школьникам можно предложить ознакомиться с руководством [10] и рассмотреть задачи, предложенные в [1, 11].

4. Задания на анализ и изменение готовой программы

Задание 4.1.

Выполните код, расположенный в первом столбце таблицы 3, изучите комментарии, расположенные во втором столбце.

Задание 4.2.

Измените код программы из задания 4.1 так, чтобы:
а) в заголовке окна были указаны ваши фамилия и имя;
б) ширина окна была в два раза больше, чем высота;
в) окно было расположено в правом нижнем углу экрана.

Задание 4.3.

До команды `window.show()` добавьте команды:

```
lb=QLabel(window)
lb.setText('I can add text')
lb.setGeometry(10, 15, 100, 15)
```

Выполните программу. Какой объект появился в окне приложения?

Задание 4.4.

Измените положение надписи так, чтобы она находилась в центре окна.

Задание 4.5.

Добавьте еще одну надпись, расположите ее внизу окна по центру.

5. Задания на анализ внешнего вида готового приложения и создание аналога этого приложения

Создайте приложение, предназначенное для закрепления таблицы умножения учащимися начальной школы.

Задание 5.1.

На рисунке 1 представлен интерфейс тренажера устного счета.

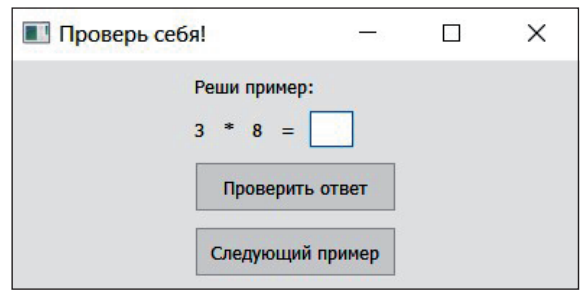

Рис. 1. «Тренажер устного счета»

Таблица 3

Текст программы	Комментарии
`from PyQt5.QtWidgets import *`	Импорт команд модуля *PyQt5.QtWidgets*
`import sys`	Подключение модуля *sys*
`app = QApplication([])`	Создание приложения
`window = QWidget()`	Создание окна
`window.setWindowTitle("Первая программа")`	Задание заголовка окна
`window.setGeometry(100, 150, 800, 370)`	Задание положения и размеров окна
`window.show()`	Отображение окна
`sys.exit(app.exec_())`	Запуск цикла обработки событий

1) Компоненты каких видов использованы для создания интерфейса?
2) Какие из текстовых полей являются константами?
3) Какие из текстовых полей должны заполняться случайными числами?
4) Какие действия должны происходить при нажатии на каждую из кнопок?

Задание 5.2.

Создайте окно и расположите на нем элементы, как показано на рисунке 1.

Решение представлено в листинге 2.

Задание 5.3.

Напишите функцию, которая очищает текстовое поле для ввода ответа и надпись для вывода результата проверки, а также размещает в надписях случайные числа от 1 до 10. Напишите команду для обработки сигнала нажатия на кнопку «Следующий пример».

Замечание: для очистки поля можно использовать пустую строку (пустые кавычки), для задания случайного числа в диапазоне от 1 до 9 используйте функцию *randint*(1,10) (необходимо подключить модуль *random*).

Решение представлено в листинге 3.

```
#Подключение модулей
import sys
from PyQt5.QtWidgets import *

#Создание приложения и окна
app = QApplication([])
window = QWidget()
window.setWindowTitle("Проверь себя!")
window.setGeometry(300, 250, 320, 270)

# Размещение в окне надписи с текстом задания
task = QLabel(window)
task.setText('Реши пример: ')
task.setGeometry(80, 25, 120, 25)

# Размещение в окне надписи для первого множителя
fact1 = QLabel(window)
fact1.setGeometry(80, 65, 20, 25)

# Размещение в окне надписи со знаком "*"
op = QLabel(window)
op.setText('*')
op.setGeometry(100, 65, 20, 25)

# Размещение в окне надписи для второго множителя
fact2 = QLabel(window)
fact2.setGeometry(120, 65, 20, 25)

# Размещение в окне надписи со знаком "="
eq = QLabel(window)
eq.setText('=')
eq.setGeometry(140, 65, 20, 25)

# Размещение в окне текстового поля для ответа
answer=QLineEdit(window)
answer.setGeometry(160, 65, 30, 25)

# Размещение в окне надписи для оценки верности ответа
correct = QLabel(window)
correct.setGeometry(220, 65, 60, 25)

# Размещение в окне кнопки для запуска функции проверки
bt_check=QPushButton("Проверить ответ", window)
bt_check.setGeometry(80, 115, 140, 35)

# Размещение в окне кнопки для запуска следующего примера
bt_next=QPushButton("Следующий пример", window)
bt_next.setGeometry(80, 165, 140, 35)

# отображение окна и запуск цикла обработки событий
window.show()
sys.exit(app.exec_())
```

Листинг 2

```
from random import*
def begin():
    correct.setText('')
    answer.setText('')
    fact1.setNum(randint(1, 10))
    fact2.setNum(randint(1, 10))
bt_next.clicked.connect(begin)
```

Листинг 3

```
def check_start():
    if int(fact1.text())*int(fact2.text())==int(answer.text()):
        correct.setText('Верно:)')
    else:
        correct.setText('Неверно:(')
bt_check.clicked.connect(check_start)
```

Листинг 4

```
from random import*
import sys
from PyQt5.QtWidgets import *

def go():
    bt.setGeometry(randint(1, 450),randint(1, 450), 140, 35)

app = QApplication([])
window = QWidget()
window.setWindowTitle("Догонялки!")
window.setGeometry(300, 250, 500, 500)

bt=QPushButton("Бежим!", window)
bt.setGeometry(80, 115, 140, 35)
bt.clicked.connect(go)

window.show()
sys.exit(app.exec_())
```

Листинг 5

Задание 5.4.

Напишите функцию, которая проверяет правильность введенного в текстовое поле ответа и выводит результат проверки в соответствующую надпись. Напишите команду для обработки сигнала нажатия на кнопку «Проверить ответ».

Решение представлено в листинге 4.

Задание 5.5.

При запуске приложения надписи, в которых должны быть расположены множители, остаются пустыми. Исправьте это, добавив в основную программу вызов функции, описанной в задании 5.3.

6. Задания для самостоятельного выполнения

Задание 6.1. «Догонялки».

Создайте окно приложения, расположите в нем кнопку с надписью «Бежим!» (рис. 2). Кнопка, при нажатии на нее, должна перемещаться в другое место окна, определяемое случайным образом.

Решение представлено в листинге 5, ***результат работы программы*** — на рисунке 2.

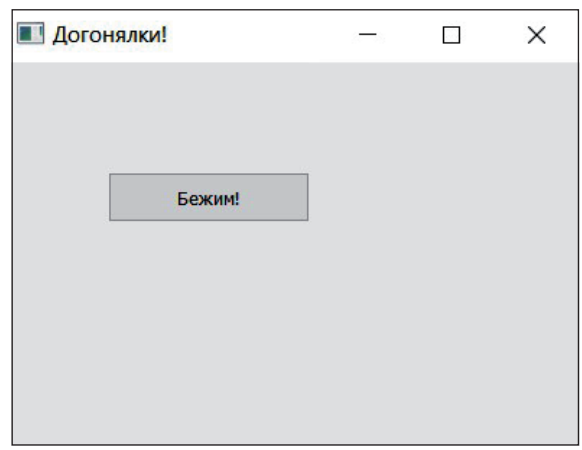

Рис. 2. «Догонялки»

Задание 6.2. «Фитнес-окно».

Создайте окно приложения, расположите в нем две кнопки с надписями «Толстеть» и «Худеть» (рис. 3). По щелчку мыши по каждой из кнопок окно должно либо растягиваться, либо сужаться. Постарайтесь сделать так, чтобы положение кнопок при этом оставалось по центру окна.

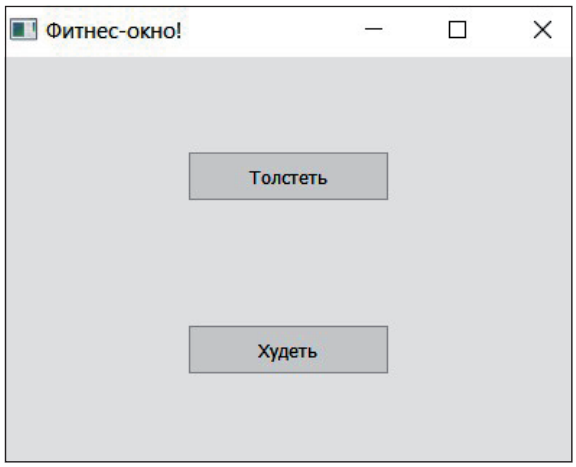

Рис. 3. «Фитнес-окно»

Решение представлено в листинге 6, ***результат работы программы*** — на рисунке 3.

Задание 6.3. «Подвижное окно».

Создайте окно приложения, расположите в нем четыре кнопки с надписями «Вверх», «Вниз», «Вправо» и «Влево» (рис. 4). По щелчку мыши по каждой из кнопок окно должно перемещаться в соответствующую сторону.

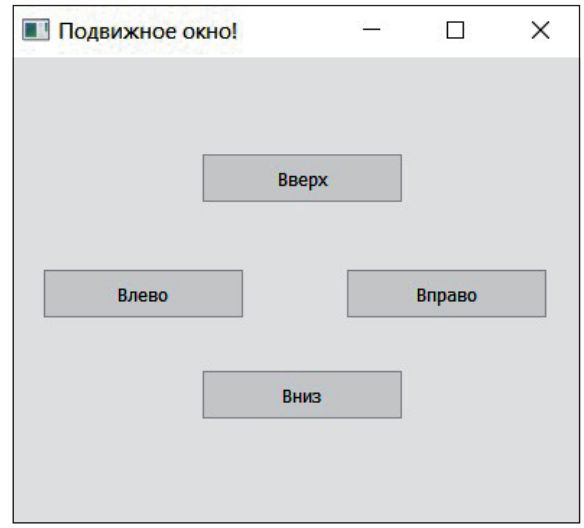

Рис. 4. «Подвижное окно»

Решение представлено в листинге 7, ***результат работы программы*** — на рисунке 4.

```
from PyQt5.QtWidgets import *
import sys

def fat():
   window.setGeometry(500, 350, window.width() + 10, 400)
   bt_fat.setGeometry(window.width() // 2 - 70, 70, 140, 35)
   bt_skinny.setGeometry(window.width() // 2 - 70, 220, 140, 35)

def skinny():
   if window.width() > bt_fat.width():
      window.setGeometry(500, 350, window.width() - 10, 400)
      bt_fat.setGeometry(window.width() // 2 - 70, 70, 140, 35)
      bt_skinny.setGeometry(window.width() // 2 - 70, 220, 140, 35)

app = QApplication ([])
window = QWidget()
window.setWindowTitle("Фитнес-окно!")
window.setGeometry(500, 350, 500, 400)

bt_fat = QPushButton("Толстеть", window)
bt_fat.setGeometry(window.width() // 2 - 70, 70, 140, 35)
bt_fat.clicked.connect(fat)

bt_skinny = QPushButton("Худеть", window)
bt_skinny.setGeometry(window.width() // 2 - 70, 220, 140, 35)
bt_skinny.clicked.connect(skinny)

window.show()
sys.exit(app.exec_())
```

Листинг 6

```
from PyQt5.QtWidgets import *
import sys

def up():
    window.move(window.x(), window.y()-10)
def down():
    window.move(window.x(), window.y() + 10)
def right():
    window.move(window.x()+10, window.y())
def left():
    window.move(window.x()-10, window.y())

app = QApplication([])
window = QWidget()
window.setWindowTitle("Подвижное окно!")
window.setGeometry(500, 350, 400, 400)

bt_up=QPushButton("Вверх", window)
bt_up.setGeometry(130, 70, 140, 35)
bt_up.clicked.connect(up)

bt_down=QPushButton("Вниз", window)
bt_down.setGeometry(130, 220, 140, 35)
bt_down.clicked.connect(down)

bt_right=QPushButton("Вправо", window)
bt_right.setGeometry(230, 150, 140, 35)
bt_right.clicked.connect(right)

bt_left=QPushButton("Влево", window)
bt_left.setGeometry(20, 150, 140, 35)
bt_left.clicked.connect(left)

window.show()
sys.exit(app.exec_())
```

Листинг 7

7. Самостоятельная разработка интерфейса и реализация приложения в соответствии с предложенными требованиями

Создайте приложение для игры, в которой компьютер загадывает случайное число, а пользователь пытается его угадать.

Алгоритм работы учащихся может быть следующим:

1) Создание окна приложения и размещение в нем надписи с правилами игры, текстового поля для ввода предполагаемого числа, надписи для вывода сообщения о результатах угадывания числа, кнопок «Новая игра» и «Проверить».
2) Разработка функции для обработки события при нажатии кнопки «Новая игра». В этой функции необходимо: объявить глобальную переменную для случайного числа от 1 до 100 и присвоить ей значение, очистить текстовое поле для ввода предполагаемого числа и надпись для вывода сообщения о результатах угадывания числа.
3) Разработка функции для обработки события при нажатии кнопки «Проверить». В этой функции необходимо написать оператор ветвления, в котором загаданное число сравнивается с содержимым текстового поля, и в зависимости от результата сравнения выводится одно из сообщений: «Число угадано! Вы выиграли!», «Загаданное число больше», «Загаданное число меньше».

Решение представлено в листинге 8.

8. Дополнительные задания

Задание 1.

Дополните программу «Угадай число», добавив счетчик попыток: если количество попыток становится больше 7, то игра заканчивается сообщением «Вы проиграли!»

Задание 2.

Дополните программу «Угадай число», добавив таймер: если пользователь не вводит ответ в течении 10 секунд, то таймер обновляется и прибавляется попытка.

```
from PyQt5.QtWidgets import *
import sys
from random import *

def start_action():          #начало новой игры
    global k
    k = randint(1, 100)
    lb_check.setText('')
    le_input.setText('')

def check_action():          #проверка введенного числа
    if int(le_input.text()) == k:
        lb_check.setText("Число угадано!\n Вы выиграли!")
    elif int(le_input.text()) < k:
        lb_check.setText("Загаданное число больше")
        le_input.clear()
    else:
        lb_check.setText("Загаданное число меньше")
        le_input.clear()

# создание приложения и окна
app = QApplication([])
window = QWidget()
window.setWindowTitle("Угадай число ")
window.setGeometry(300, 250, 400, 370)

# "загадывание" случайного числа
k = randint(1, 100)
# создание надписи с условием игры
lb_hello=QLabel(window)
lb_hello.setText('Добро пожаловать в игру "Числовая угадайка"!\nВаша задача: угадать
число, которое я загадал.')
lb_hello.setGeometry(50, 10, 550, 50)

# создание надписи с пояснением
lb_input = QLabel(window)
lb_input.setText('Введи число от 1 до 100:')
lb_input.setGeometry(50, 75, 150, 15)

# создание текстового поля для ввода угадываемого числа
le_input = QLineEdit(window)
le_input.setGeometry(50, 105, 80, 40)
le_input.setMaxLength(3)

# создание кнопки для запуска новой игры
bt_newgame = QPushButton("Новая игра", window)
bt_newgame.setGeometry(50, 210, 120, 40)
bt_newgame.clicked.connect(start_action)

#создание кнопки для проверки соответствия введенного числа загаданному
bt_check = QPushButton("Проверить", window)
bt_check.setGeometry(50, 160, 100, 40)
bt_check.clicked.connect(check_action)

# создание надписи для вывода результата проверки соответствия введенного числа
загаданному
lb_check = QLabel(window)
lb_check.setGeometry(155, 105, 160, 40)
lb_check.setText('')

window.show()
sys.exit(app.exec_())
```

Листинг 8

В архиве дополнительных материалов к данной статье на сайте ИНФО* можно найти решения указанных задач и справочный материал с описанием методов для класса *QTimer*.

9. Заключение

Последовательность представленных выше заданий построена в соответствии с возрастанием уровня субъектности учащихся: от работы с готовой моделью до построения своих моделей, что способствует развитию таких составляющих вычислительного мышления, как декомпозиция, оценка, сопоставление с образцом и выбор правильного представления данных для решения рассматриваемой задачи [2, 3].

Практика показывает, что указанный набор заданий позволяет школьникам, получив минимальный объем справочной информации и опыт ее использования, применять полученные навыки при решении других задач.

Список источников

1. *Бабушкина И. А., Окулов С. М.* Практикум по объектно-ориентированному программированию: учебное пособие. М.: БИНОМ. Лаборатория знаний, 2012. 372 с. EDN: RBATTD.

2. *Босова Л. Л.* Как учат программированию в XXI веке: отечественный и зарубежный опыт обучения программированию в школе // Информатика в школе. 2018. № 6. С. 3–11. EDN: XZOOJV.

3. *Босова Л. Л.* Международные тенденции развития школьного образования в области информатики и ИКТ // Материалы Шестнадцатой открытой Всероссийской конференции «Преподавание информационных технологий в Российской Федерации» (Москва, 14–15 мая 2018 года). М.: МГТУ имени Н. Э. Баумана, 2018. С. 350–352. EDN: XUJFSX.

4. *Босова Л. Л., Аквилянов Н. А., Кочергин И. О., Штепа Ю. Л., Бурцева Т. А.* Информатика. 8–9 классы. Начала программирования на языке Python: дополнительные главы к учебникам. М.: БИНОМ. Лаборатория знаний, 2020. 96 с. EDN: LPMXCB.

5. *Дмитриев В. Л., Мухаметова Л. К., Евдокимова Н. Л.* Игровые проекты в школьном курсе программирования // Информатика в школе. 2016. № 2. С. 49–54. EDN: VVQWGN.

6. *Заика И. В.* Работа динамическими массивами и текстовыми файлами в визуальной среде разработки программ Delphi // Информатика в школе. 2020. № 4. С. 44–48. EDN: FGGJOF. DOI: 10.32517/2221-1993-2020-19-4-44-48.

7. *Каракозов С. Д., Маняхина В. Г.* Python как базовый язык обучения программированию // Информатика в школе. 2020. № 1. С. 26–30. EDN: GPEZPH. DOI: 10.32517/2221-1993-2020-19-1-26-30.

8. *Поляков К. Ю., Еремин Е. А.* Информатика. 11 класс. Базовый и углубленный уровни. В 2 ч. Ч. 2. М.: БИНОМ. Лаборатория знаний, 2017. 304 с.

9. *Поляков К. Ю., Еремин Е. А.* Новая линейка учебников информатики для основной и средней школы // Актуальные проблемы физико-математического образования. Материалы II Международной научно-практической конференции (Набережные Челны, 20–22 октября 2017 года). Набережные Челны: Набережночелнинский государственный педагогический университет, 2017. С. 233–234. EDN: YUXTFB.

10. *Прохоренок Н. А., Дронов В. А.* Python 3 и PyQt 5. Разработка приложений. СПб.: БХВ-Петербург, 2018. 832 с.

11. *Симонович С. В., Евсеев Г. А.* Занимательное программирование: Delphi: книга для детей, родителей и учителей. М.: Аст-ПрессКнига; Информком-Пресс, 2003. 368 с. EDN: OZNWUI.

12. Федеральная рабочая программа среднего общего образования. Информатика (углубленный уровень) (для 10–11 классов образовательных организаций). https://edsoo.ru/wp-content/uploads/2023/08/22_ФРП_Информатика-10-11-классы_угл.pdf

* http://infojournal.ru/journals/school/school_04-2024/