https://doi.org/10.53656/math2024-6-5-usi

ИЗПОЛЗВАНЕ НА VISUAL BASIC FOR APPLICATIONS В EXCEL ЗА РАЗВИТИЕ НА УМЕНИЯТА НА УЧЕНИЦИТЕ В ОБЛАСТТА НА ПРОГРАМИРАНЕТО

Стефка Анева, Елена Тодорова
Пловдивски университет „Паисий Хилендарски“ – Пловдив (България)

Резюме. В настоящата статия е представен един подход за развитие и усъвършенстване на практическите и приложните умения на учениците в областта на програмирането, с използване на възможностите на Visual Basic for Applications в Excel. Представени са пет задачи и техните решения, реализирани с помощта на вградените средства за програмиране в Excel. Чрез тези примери се автоматизира изпълнението на следните дейности – определяне на броя на четни и нечетни числа от краен брой елементи и извличане на съответните списъци; намиране на всички прости числа в зададен интервал; сливане на два сортирани едномерни масива с определен брой елементи в трети подреден масив; познаване на случайно генерирано цяло число в зададен интервал. С предложения подход и задачи се цели прилагането на формираните знания и умения на учениците в областта на програмирането в гимназиален етап, свързани с използване на разклонени и циклични алгоритмични конструкции, работа с едномерни масиви, както и креативно използване на формираните дигитални умения на учениците за обработка на таблични данни.

Ключови думи: информатика; информационни технологии; алгоритмични умения; алгоритми; Visual Basic for Applications; електронни таблици; дигитални умения; дигитална креативност

1. Въведение

Процесът на формиране на знания и умения у учениците в областта на програмирането започва още в началното училище и се надгражда и развива във всеки следващ етап на обучение. В обучението по учебния предмет компютърно моделиране в началното училище учениците формират базови алгоритмични знания и практически и приложни умения за конструиране на линейни, разклонени и циклични алгоритми при работа с конкретна визуална среда за блоково програмиране. В обучението по учебния предмет компютърно моделиране и информационни технологии (КМИТ) в прогимназията, при изучаване на темата „Компютърно моделиране“ се усвояват теоретични знания за различни езици за програмиране и се формират практически умения за използването им при ре-

шаването на задачи с различна степен на сложност (Aneva & Todorova 2021).

При изучаването на предмета информатика[1] в гимназиалния етап на обучение се засягат теми, свързани с основни области на компетентност, като обектно-ориентирано програмиране, графичен потребителски интерфейс, алгоритми и структури от данни и др.

Решаването на задачи с компютър чрез програмиране на конкретен език е основна дейност на учениците в обучението по информатика в училище (Dureva 2003). Съставянето на компютърни програми на различни езици за програмиране и непосредствената работа с различни програмни среди реализират специфични дидактически цели като: формиране на знания и практически умения за работа с конкретна среда за програмиране, прилагане на различни алгоритми за решаване на реални практически задачи чрез използване на различни програмни езици, творческо прилагане на усвоените знания и умения за решаването на проблеми с компютър и др. (Garov 2010; Rahnev & Garov 1988; Rahnev & Garov 1988a; Rahnev et al. 1990; Grozdev & Garov 2008).

В гимназиален етап на обучение формираните у учениците, в началното училище и в прогимназията, базови знания и умения в областта на програмирането могат да бъдат успешно приложени и развити при работа с други основни програмни езици като Python, JavaScript, Java, C#, Visual Basic и др. (Manev et al. 2017; Garov 2013).

Съществен момент в процеса на обучение е усвояването на знанията и уменията да се отличава с трайност и задълбоченост, учениците да могат да възпроизвеждат и използват в различни ситуации съществените от изучения учебен материал, свързан с програмирането, както и да прилагат вече формираните знания и умения в нов контекст при използване на друг програмен език и среда за програмиране (Todorova et al. 2021).

В (Aneva & Todorova 2024) разгледахме един подход за креативно използване на формираните дигитални умения на учениците за обработка на таблични данни, както и на техните умения в областта на програмирането при реализацията на познати за тях алгоритми от обучението по КМИТ в 5. клас, прилагайки ги в нов контекст и среда за програмиране, а именно VBA в Excel. Предложеният подход предоставя възможности за реализиране на интегрирано вътрешнопредметно взаимодействие в обучението по КМИТ в прогимназията. Предложеният подход предоставя възможности за реализиране на интегрирано вътрешнопредметно взаимодействие в обучението по КМИТ в прогимназията и може да бъде приложен в часовете за разширена и допълнителна подготовка по предмета, както и при провеждане на извънкласни дейности и занимания по интереси.

2. Реализации на някои алгоритми с вградените средства за програмиране в Excel

В настоящата работа представяме примери за креативно използване на формираните алгоритмични знания и умения на учениците за прилагане на линейни, разклонени и циклични конструкции, формирани в прогимназиален етап, при реализацията на някои алгоритми в обучението по програмиране в гимназиален етап и прилагането на тези знания в нов контекст и среда за програмиране. За целта ще използваме възможностите на Excel и вградените средства за програмиране в тази среда, предоставени от Visual Basic for Applications (VBA), и ще разгледаме примерни идейни реализации на някои алгоритми, свързани с търсене и броене на елементи, подреждане на елементи по големина, както и работа с едномерни масиви (деклариране, инициализиране, въвеждане и извеждане на елементи на масив, сливане на подредени масиви и др.). Ще разгледаме пет задачи, свързани с реализацията на следните дейности:

• търсене и преброяване на нечетни и четни елементи за краен брой въведени целочислени данни и тяхното директно извличане в последователно разположени една под друга клетки в работен лист;

• търсене и преброяване на нечетни и четни елементи за краен брой въведени целочислени данни, извличането им в два едномерни масива и разполагане на получените резултати в последователно разположени в една колона клетки в работен лист;

• търсене и преброяване на всички прости числа в зададен интервал;

• сливане на елементите на два сортирани едномерни масива в трети сортиран масив;

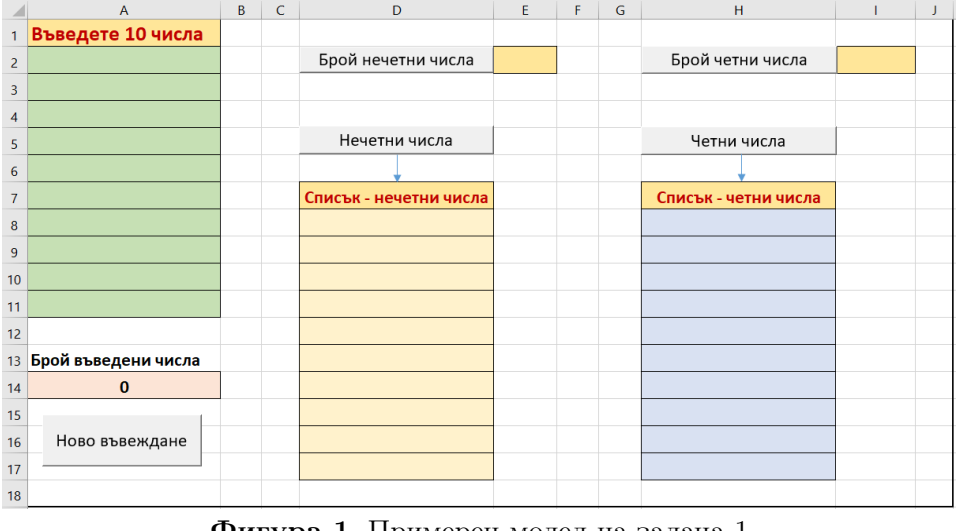
• „познаване“ на случайно генерирано цяло число в зададен интервал.

Представеният подход и задачи могат да намерят приложение при организиране и провеждане на извънкласни форми на обучение по информатика и/или информационни технологии в гимназиален етап. По този начин учениците биха усъвършенствали своите алгоритмични знания и умения за използване на условни и циклични програмни конструкции при реализация на различни алгоритми, дефиниране на локални и глобални променливи, използване на различни типове данни, управление на входните данни, дефиниране и използване на подпрограми, работа с масиви. Същевременно ще се запознаят с различни езици и среди за програмиране, както и средства за създаване и изпълнение на програмен код.

От друга страна, спрямо актуалната (действаща към момента) учебна програма по информационни технологии за 10. клас[2] (общообразователна подготовка), при изучаване на темата „Техническа и организационна сигурност при работа в дигитална среда“, учениците се запознават с предназначението на макросите в офис приложенията и формират умения да

управляват включването им при използване на публични услуги. В този аспект, в процеса на обучението по темата може да бъде разгледана накратко и самата технология за създаване на макроси в среда на Excel чрез средствата на VBA. По този начин, използвайки вградените средства за програмиране в Excel, учениците биха интегрирали своите умения както за обработка на таблични данни (например за именуване на области, валидиране на данни, условно форматиране и други), така и в областта на програмирането и да ги приложат съвместно при разрешаването на конкретни практически проблеми.

Задача 1. Със средствата на Excel и VBA реализирайте дейностите по броене на нечетни и четни елементи за въведени данни в десет клетки в работен лист на Excel и формиране на отделни списъци с нечетните и четните елементи в последователно разположени една под друга клетки в тсъщия работен лист.



Фигура 1. Примерен модел на задача 1

Решение: В работен лист на Excel се реализира конкретният модел на задачата по посочения образец на фиг. 1. За въвеждане на десетте стойности ще се използват последователно разположени клетки в работния лист – в случая това е областта A2:A11. След извършване на преброяване на нечетните и четните елементи, крайния резултат – съответният брой – ще се визуализира в клетките E2 и I2. Отделните списъци с нечетни и четни елементи ще се представят в следните области от клетки – D8:D17 и H8:H17. Всяка една от тези области е съставена от по десет

клетки, тъй като въведените десет числа могат да бъдат както само нечетни, така и само четни. Работният лист в Excel, в който е реализиран моделът на задачата, е именуван като обект във VBA по следния начин – Sheet_OddEven.

Следващ етап от решаването на задачата е реализирането на основните функционалности с помощта на VBA и създаването на съответните макроси, автоматизиращи следните дейности – *определяне на броя на нечетните и четните елементи за въведени стойности в десет клетки; формиране на съответните списъци; подготовка за ново въвеждане.* Допълнително към задачата може да се реализира проверка за пълнота на въведени числови данни за десетте клетки и след това тези данни да бъдат подложени на преброяване относно нечетни и четни елементи. За улесняване на дейността по въвеждане на данни за десетте клетки може да се реализира автоматично попълване чрез използване на случайно генериране на числа в зададен интервал (например в интервала [1,100]).

За автоматизиране на дейността по определяне на броя на нечетните и четните елементи за въведените десет стойности в случая е представена концепция за създаване на отделни макроси, но може да бъде реализирана и една обща подпрограма, извършваща едновременно и двете преброявания.

При реализиране на подпрограмата за определяне на броя на нечетните елементи може да се подходи по два начина:

• чрез използване на оператор за цикъл `For` с управляваща променлива `i`, с който се обхождат отделните елементи (въведени в десетте клетки в колона `A` и ред `i+1`, като `i` приема стойности от `1` до `10`), за които се извършва проверка дали са нечетни и техният брой се натрупва в променливата `oddnum` (листинг 1);

• чрез използване на оператор за цикъл `For Each`, с който се обхожда всяка клетка от областта `A2:A11`, за която се извършва проверка дали нейната стойност е нечетно число и броят на клетките, съдържащи нечетни числа, се натрупва в променливата `oddnum` (листинг 2).

Аналогично може да се подходи и при определяне на броя на четните елементи.

Листинг 1. Определяне броя на нечетни елементи за въведени данни в десет клетки (*чрез използване на оператор за цикъл* `For`)

```
Public Sub br_nechetni_chisla_v1()
  Dim oddnum As Integer
  For i = 1 To 10
    If (Sheet_OddEven.Cells(i + 1, 1).Value Mod 2) = 1 _
      Then oddnum = oddnum + 1
```

```
  Next i
  Sheet_OddEven.Range("E2").Value = oddnum
End Sub
```

Листинг 2. Определяне броя на нечетни елементи за въведени данни в десет клетки *(чрез използване на* `For Each`)

```
Public Sub br_nechetni_chisla_v2()
  Dim oddnum As Integer
  For Each xcell In Sheet_OddEven.Range("A2:A11")
    If xcell Mod 2 = 1 Then oddnum = oddnum + 1
  Next xcell
  Sheet_OddEven.Range("E2").Value = oddnum
End Sub
```

Листинг 3. Извличане на нечетните елементи и формиране на отделен списък в областта D8:D17

```
Public Sub show_Oddnums()
  Dim num As Integer
  Sheet_OddEven.Range("D8:D17").ClearContents
  br_nechetni_chisla_v1
  'num - начален ред за разполагане на резултата
  num = 8
  For i = 1 To 10
    If (Sheet_OddEven.Cells(i + 1, 1).Value Mod 2) = 1 Then
      Sheet_OddEven.Cells(num, 4).Value = _
        Sheet_OddEven.Cells(i + 1, 1).Value
      num = num + 1
    End If
  Next i
End Sub
```

При извличане на четните елементи и тяхното попълване в отделен списък може да се подходи както в листинг 3, като се има в предвид, че в случая областта за визуализиране на списъка ще бъде H8:H17.

Листинг 4. Подготовка за ново въвеждане на данни

```
Public Sub new_data()
  Sheet_OddEven.Range("A2:A11").ClearContents
  Sheet_OddEven.Range("E2,I2").ClearContents
  Sheet_OddEven.Range("D8:D17").ClearContents
  Sheet_OddEven.Range("H8:H17").ClearContents
End Sub
```

За да се провери дали са въведени данни в десетте клетки, може да се подходи по следния начин: в клетка A14 на работния лист в Excel с модела на задачата, с помощта на функция `COUNT`, да се преброят въведените числови стойности в областта A2:A11. След това във вече реализираните макроси за преброяване на нечетни и четни елементи с помощта на променлива `br` се прочита стойността на клетка A14 и се извършва проверка дали въведените числа са 10 на брой, и едва след това се извършват останалите дейности. Примерен вариант за надграждане на кода за макроса br_nechetni_chisla_v1 е представен в листинг 5. Аналогично може да се подходи при написване и на останалите процедури.

Листинг 5. Надграждане на програмния код от **Листинг 1** (*при определяне на броя на нечетни елементи*) с добавена проверка за пълнота на въведените данни за десетте клетки

```
Public Sub br_nechetni_chisla_v1()
  Dim oddnum, br As Integer
  'въвеждане на формула в клетка A14 в среда на VBA
  Sheet_OddEven.Range("A14").Formula = "=COUNT(A2:A11)"
  br = Sheet_OddEven.Range("A14").Value
  If br = 10 Then
    For i = 1 To 10
      If (Sheet_OddEven.Cells(i + 1, 1).Value Mod 2) = 1 _
        Then oddnum = oddnum + 1
    Next i
    Sheet_OddEven.Range("E2").Value = oddnum
  Else
    MsgBox "Непълни данни!"
  End If
End Sub
```

При надграждане на програмния код за определяне броя на четните елементи (с добавена проверка за пълнота на въведените данни за десетте клетки) може да се подходи аналогично, като полученият в променливата `evennum` брой на четните елементи ще се визуализира в клетка I2.

Задача 2. Със средствата на Excel и VBA реализирайте дейността по броене на нечетни и четни елементи за въведени данни в десет клетки в работен лист на Excel. В процеса на извличане на съответните нечетни и четни елементи формирайте два едномерни масива `Odd` (*за добавяне на нечетните елементи*) и `Even` (*за добавяне на четните елементи*).

Елементите на получените едномерни масиви с нечетни и четни целочислени стойности визуализирайте в две отделни области от последователно разположени една под друга клетки в същия работен лист.

Решение: За решаването на Задача 2 правим следните промени в решението на Задача 1. Най-напред декларираме на глобално ниво два динамични едномерни масива `Odd` и `Even` и две променливи `oddnum` и `evennum`. В случая ще използваме динамични масиви, тъй като не се знае точният брой нечетни и четни числа, които ще бъдат въведени в десетте клетки. Параметрите на динамичните масиви не се определят при тяхното деклариране и трябва да бъдат определени (променени) с помощта на оператор `Redim`. Типът на елементите може да се променя само когато един масив е стойност на променлива от тип `Variant`, както следва:

```
Dim Odd, Even As Variant
Dim oddnum, evennum As Integer
```

При създаването на процедурата `br_nechetni_chisla_zad2`, реализираща дейността по определяне броя на нечетните елементи, може да се използва вече реализираната в Задача 1 процедура (листинг 5), като се вземе предвид, че променливата `oddnum` вече е декларирана като глобална променлива и в процедурата само ще ѝ дадем начална стойност – т.е. `oddnum = 0`. Аналогично може да се постъпи и при определяне броя на четните елементи.

Листинг 6. Формиране на масива `Odd` от нечетни елементи

```
Sub read_arrayOdd()
  br_nechetni_chisla_zad2
  If oddnum > 0 Then
    ReDim Odd(0 To oddnum - 1) As Integer
    n = 0
    For i = 1 To 10
      If (Sheet_OddEven.Cells(i + 1, 1).Value Mod 2) = 1 _
      Then
        Odd(n) = Sheet_OddEven.Cells(i + 1, 1).Value
        n = n + 1
      End If
    Next i
  End If
End Sub
```

Аналогично може да се подходи при реализирането на процедурата `read_arrayEven()` за формиране на масива `Even` от четни елементи, като се уточнят параметрите на динамичния масив `Even`:

```
ReDim Even(0 To evennum - 1) As Integer.
```

За реализиране на дейностите по попълване на списъците с нечетни и четни елементи чрез извличане на елементите на двата едномерни масива може да се подходи по два начина:

• чрез използване на цикъл `For` за последователно обхождане на отделните елементи на съответния масив (`Odd` или `Even`) и въвеждане на неговите данни като стойности на последователно разположени клетки в областта D8:D17 (за формиране на списъка с нечетните елементи) или H8:H17 (за формиране на списъка с четни елементи). В случая тези две области включват по 10 клетки, но реално ще се попълнят последователно само толкова от тях, колкото е броят на намерените нечетни или четни елементи;

• чрез използване на метода `Transpose` и директно разполагане на елементите на конкретния масив (`Odd` или `Even`) в зададена вертикална област от клетки. Тази област трябва да бъде съставена от същия брой клетки, колкото са и елементите на съответния масив. За целта чрез променлива `s_range` от тип `String` се конфигурира областта за съответния тип елементи по следния начин:

  ○ за нечетните: `s_range = "D8:D"+ Format(num + oddnum - 1);`

  ○ за четните: `s_range = "H8:H"+ Format(num + evennum - 1),`

където `num` е началният ред за разполагане на получените списъци (в случая `num = 8`).

Листинг 7. Извличане на елементите на масива `Odd` и формиране на отделен списък с нечетните елементи (*първи вариант*)

```
Public Sub show_Oddnums_zad2()
  Dim num As Integer
  Sheet_OddEven.Range("D8:D17").ClearContents
  'num -- начален ред за разполагане на резултата
  num = 8
  read_arrayOdd
  For i = 0 To oddnum - 1
    Sheet_OddEven.Cells(num, 4).Value = Odd(i)
    num = num + 1
  Next i
End Sub
```

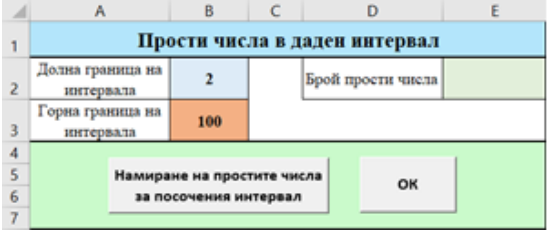
Листинг 8. Извличане на елементите на масива `Odd` и формиране на отделен списък с нечетните елементи (*втори вариант*)

```
Public Sub show_Oddnums_zad2_v2()
  Dim num As Integer
```

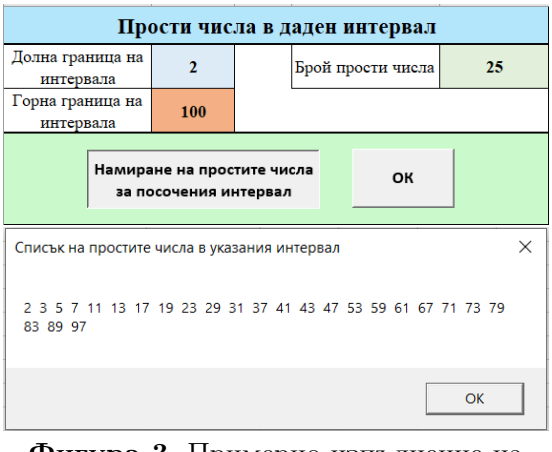
```
  Dim s_range As String
  Sheet_OddEven.Range("D8:D17").ClearContents
  num = 8
  read_arrayOdd
  If oddnum > 0 Then
    s_range = "D8:D" + Format(num + oddnum - 1)
    Sheet_OddEven.Range(s_range).Value = _
      Application.Transpose(Odd)
  End If
End Sub
```

Задача 3. Със средствата на Excel и VBA реализирайте търсене, преброяване и извличане на всички прости числа в указан интервал.

Решение: В работен лист на Excel реализираме конкретния модел на задачата по посочения примерен образец на фиг. 2. За въвеждане на долната и горната граница на интервала ще използваме последователно разположени клетки от работен лист в Excel – в случая това са клетки B2 и B3. За тези клетки с помощта на командата Data Validation може да се зададе валидиране при въвеждане на данните: за клетка B2 – цели числа, по-големи от 1, а за клетка B3 – цели числа, по-големи от стойността на B2. След намирането на броя на простите числа в зададения интервал крайният резултат ще се визуализира в клетка E2. Списъкът с намерените елементи, разделени с по един интервал, ще се натрупва в променлива `s` от тип `String`, а крайният резултат ще се визуализира с помощта на диалогова кутия за съобщения MsgBox.



Фигура 2. Примерен модел на задача 3



Фигура 3. Примерно изпълнение на задача 3

Следващ етап от решаването на задачата е реализирането на основните функционалности с помощта на VBA и създаването на макроси, автоматизиращи следните дейности – *определяне на броя на всички прости числа в указания интервал и визуализиране на списъка с простите числа в диалогова кутия за съобщения, както и за изчистване на данни.*

Предложената подпрограма и алгоритъм (листинг 9) за определяне на простите числа в зададен интервал включва следните дейности:

• деклариране на необходимите локални променливи (`a`, `b`, `br`, `x`, `i`);

• в променливите `a` и `b` се прочитат въведените в клетки B2 и B3 цели числа. Тези две променливи задават интервала [`a`,`b`], в който ще се търсят прости числа;

• на променливата `br` задаваме начална стойност 0. В нея ще натрупваме броя на намерените прости числа по време на обхождането на интервала [`a`,`b`];

• прави се проверка за коректност на интервала (`a` $<$ `b` и `a` $>$ 1). В случай че в работния лист в Excel е зададена проверка за валидност на данните за клетки B2 и B3, тази проверка може да се пропусне. При коректно зададен интервал [`a`,`b`], с помощта на цикъл `For` с управляваща променлива `x` се обхождат всички числа в интервала и се извършва проверка дали са прости, или не. За всяка стойност на `x` алгоритъмът за проверка дали зададено цяло число е просто, се свежда до следното:

  ○ на променливата `i` се задава начална стойност 2;

  ○ на булевата променлива `p` се задава начална стойност `true` (т.е. предполагаме, че съдържанието на `x` е просто число);

  ○ с цикъл `While` се проверява дали стойността на `x` се дели на число в интервала от 2 до $\sqrt{x}$. Проверката се извършва до $\sqrt{x}$, защото наличието на делител, по-голям от $\sqrt{x}$, означава съществуването и на такъв, който е по-малък от $\sqrt{x}$, така че умножени, двата да дават числото. По този начин броят на проверките става много по-малък. Ако се намери делител, числото `x` не е просто и променливата `p` приема стойност `false`;

  ○ ако след приключване на проверката стойността на променливата `p` е `true`, числото е просто. В този случай стойността на променливата `br` се увеличава с 1 и в променливата `s` от тип `String` добавяме преобразуваното в низ намерено просто число, отделяйки го с един интервал от предното.

• Накрая, в клетка E2 се показва стойността на променливата `br` и се визуализира диалогова кутия за съобщения, показваща стойността на променливата `s`, в която се намира формираният списък с всички прости числа в зададения интервал.

Листинг 9. Реализиране на дейността по търсене, преброяване и извличане на всички прости числа в указан интервал

```
Sub prosti_chisla()
  Dim a, b, br, x, i As Integer
  Dim p As Boolean: Dim s As String
  a = Int(Sheet1.Range("B2").Value)
  b = Int(Sheet1.Range("B3").Value)
  br = 0
  If a < b And a > 1 Then
    For x = a To b
      i = 2: p = True
      While (i <= Sqr(x) And p)
        If x Mod i = 0 Then p = False
        i = i + 1
      Wend
      If p Then
        br = br + 1: s = s + Str(x) + " "
      End If
    Next x
    Sheet1.Range("E2").Value = br
    MsgBox s, 0, "Списък на простите числа в указания интервал"
  Else
    MsgBox "Въведете коректен интервал [a,b], a > 1!"
  End If
End Sub
```

При реализиране на дейността по изчистване на данните и подготовка за ново въвеждане може да се подходи както в Задача 1, като в случая трябва да се изчистят данните в клетки B2, B3 и E2.

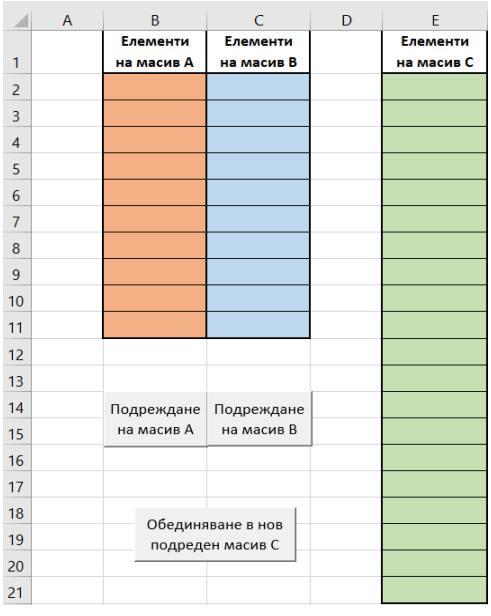
Задача 4. Със средствата на Excel и VBA реализирайте дейността по сливане на елементите на два подредени във възходящ ред едномерни масива A и B (съдържащи по десет елемента) в трети подреден едномерен масив C.

Решение: В работен лист на Excel се реализира конкретният модел на задачата по посочения примерен образец от фиг. 4. За въвеждане на целочислените стойности за елементите на двата едномерни масива A и B ще се използват последователно разположени клетки от работен лист в Excel – в случая това са клетките в колони B и C. Използвайки технологията за именуване на области в Excel, ще зададем следните имена на области:

• `range_arrayA` – за областта от клетки B2:B11, съдържащи елементите на масива `A`;

• `range_arrayB` – за областта от клетки C2:C11, съдържащи елементите на масива `B`;

• `range_arrayC` – за областта от клетки E2:E21, където ще се визуализират елементите на обединения подреден масив `C`.



Фигура 4. Примерен модел на задача 4

След извършване на дейностите по подреждане във възходящ ред на елементите на двата масива `A` и `B`, както и обединяване на двата сортирани масива в масива `C`, крайният резултат от обединението ще се визуализира в областта E2:E21.

Следващият етап от решението на задачата е реализирането на основните функционалности с помощта на VBA и създаването на съответните макроси, автоматизиращи следните дейности – *подреждане на елементите на масива* `A` *във възходящ ред, подреждане на елементите на масив* `B` *във възходящ ред и обединяване в нов масив* `C` *(подреден във възходящ ред)*. Допълнително към задачата може да се реализира проверка за пълнота на въведени числови данни за десетте клетки в колона `B` и колона `C` и след това тези данни да бъдат подложени на подреждане по големина и обединяване в трети подреден масив.

За разлика от Задача 2, където показахме деклариране на динамичен масив и последващо определяне (променяне) на неговите параметри с използване на оператора `Redim`, в случая ще използваме статични масиви. За двата масива `A` и `B` е известно, че ще бъдат с фиксиран размер и ще съдържат точно по десет елемента. Параметрите на статичните (фиксирани) масиви не могат да се променят. Декларираме двата едномерни масива `A` и `B` като глобални:

```
Dim A(10) As Integer
Dim B(10) As Integer
```

За да сортираме елементите на двата масива `A` и `B`, използваме метода на „мехурчето“. При този метод за сортиране на редица от числа се сравняват стойностите на всеки два съседни елемента и се разменят местата

им, ако са неподходящо подредени. Ако сортираме във възходящ ред, след първото обхождане на масива най-големият елемент се позиционира на последно място и имитира „изплуване“ на мехурче в съд с течност. Процедурата се повтаря за останалите елементи.

Листинг 10. Сортиране на едномерния масив `A` във възходящ ред с метода на „мехурчето“

```
Public Sub bublesort_arrA()
  Dim br, n As Integer
  br = 0
  For Each Cell In range("range_arrayA")
    A(br) = Cell.Value: br = br + 1
  Next Cell
  n = UBound(A)
  For i = 1 To n - 1
    For j = n - 1 To i Step -1
      If A(j) < A(j - 1) Then
        P = A(j): A(j) = A(j - 1): A(j - 1) = P
      End If
    Next j
  Next i
  range("range_arrayA").Value = Application.Transpose(A)
End Sub
```

Аналогично постъпваме, за да сортираме масива `B`, като в случая използваме именуваната област `range_arrayB`.

Дейността по сливане на двата подредени (във възходящ ред) масива `A` и `B` в трети подреден масив `C` се състои в следното:

- в променливите `m` и `n` съхраняваме броя на елементите на двата масива `A` и `B` (в примера те ще съдържат по 10 елемента);
- използваме три променливи `i, j, k` – `i` и `j` за обхождане на `A` и `B`, а `k` за попълване на елементите на масива C;
- с цикъл `While`, докато е изпълнено условието `i < m And j < n`, се извършва поетапно добавяне в масива `C` на елемент от единия от двата масива, като в случай че условието `A(i) <= B(j)` е изпълнено, в масива `C` ще се добави `i`-тият елемент на масива `A`, а в противен случай в масива `C` ще се добави `j`-тият елемент на масива `B`. Цикълът `While` ще приключи, когато са обработени всички елементи на единия от двата масива;
- след това с условен оператор `If`, с условие `i < m`, се проверява дали масивът `A` е този, в който не всички елементи са обработени. Ако условието е изпълнено, тогава с цикъл `For` останалите елементи на масива `A` се добавят в масива `C`. Ако условието не е изпълнено, тогава има останали елементи в масива `B` и с цикъл `For` те се добавят в масива C.

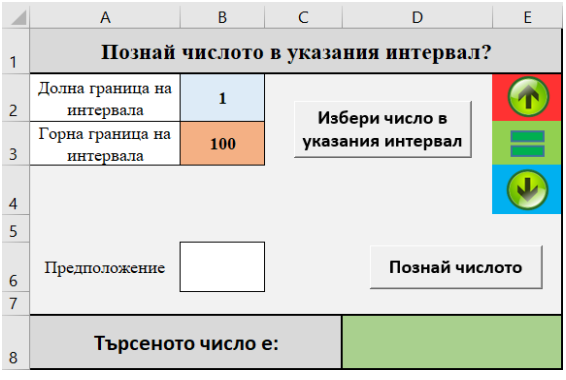
Листинг 11. Сливане на подредените масиви `A` и `B` в подреден масив `C`

```
Sub MergeTwoArrays()
  Dim i, j, k, m, n As Integer
  Dim C(20) As Integer
  bublesort_arrA: bublesort_arrB
  m = UBound(A)
  n = UBound(B)
  i = 0: j = 0: k = 0
  While (i < m And j < n)
    If A(i) <= B(j) Then
      C(k) = A(i): k = k + 1: i = i + 1
    Else
      C(k) = B(j): k = k + 1: j = j + 1
    End If
  Wend
  If (i < m) Then
    For q = i To m
      C(k) = A(q): k = k + 1
    Next q
  Else
    For q = j To n
      C(k) = B(q): k = k + 1
    Next q
  End If
  range("range_arrayC").Value = Application.Transpose(C)
End Sub
```

Допълнение: Ако в задачата се изискваше да бъдат въведени зададен от потребителя брой стойности в колони B и C на работния лист в Excel, тогава двата масива `A` и `B` трябва да бъдат с променлив брой елементи. В този случай може да се подходи аналогично на направеното в Задача 2, като тези масиви се декларират първоначално като динамични и след като се определи броят на въведените числови стойности в колони B и C, тогава да се уточнят съответните параметри на масивите с `Redim`. Същото се отнася и за масива `C`, тъй като броят на елементите му ще бъде сумарна стойност от броя елементи на двата масива `A` и `B`. Тъй като областите от клетки, в които се разполагат трите масива, ще включват променлив брой клетки, то не е удачно да се извърши именуване на тези области от клетки в средата на Excel, а трябва да се определят в средата на VBA с помощта на три променливи от тип `String` (аналогично на направеното в Задача 2).

Задача 5. Със средствата на Excel и VBA реализирайте дейността по „отгатване“ на случайно генерирано цяло число в зададен интервал.

Решение: В работен лист на Excel се реализира конкретният модел на задачата по посочения на фиг. 5 примерен образец. За въвеждане на долната и горната граница на желания интервал (в който ще се генерира случайно число) ще се използват клетки B2 и B3. В клетка B6 ще се въвеждат предположенията за числото.



Фигура 5. Примерен модел на задача 5

Следващият етап от решаването на задачата е реализирането на основните функционалности с помощта на VBA и създаването на съответните макроси, автоматизиращи следните дейности – *избор на число в указания интервал чрез използване на случайно генериране на неговата стойност; проверка дали въведеното предположение е по-голямо, по-малко или равно на това число.*

Листинг 12. Генериране на число в зададен интервал [a,b]

```
Dim x As Integer ’глобална променлива
Sub random_chislo()
  Dim a, b As Integer
  a = Range("B2").Value
  b = Range("B3").Value
  x = Int((b - a + 1) * Rnd + a)
  Range("B6").ClearContents
  ’за фон на клетка B6 се задава бял цвят
  Range("B6").Interior.Color = RGB(255, 255, 255)
  ’или клетка B6 да е без фон
  ’Range("B6").Interior.Pattern = xlNone
  Range("A8").Value = "Търсеното число е:"
  Range("D8").Value = ""
End Sub
```

За реализиране на дейността по отгатване на числото може да се подходи по следния начин:

• в променлива n се прочита стойността от клетка B6, в която потребителят въвежда своето предположение за числото;

• с помощта на блоков (структурен) условен оператор се извършва проверка дали стойността на променливата n е по-малка, по-голяма или равна на стойността на променливата x (*променливата* x *е декларирана като глобална и чрез подпрограмата* random_chislo *в нея вече е присвоена стойността на случайно генерираното число в зададения интервал*). Ако n < x, в примерното решение клетка B6 ще се оцвети в **червен цвят** и в клетка D8 ще се появи текстът по-голямо. Ако n > x, тогава клетка B6 ще се оцвети в **син цвят** и в клетка D8 ще се появи текстът по-малко. Ако n = x, клетка B6 ще се оцвети в **зелен цвят** и в клетка D8 ще се появи текстът Отгатнахте числото (фиг. 6).

Забележка. За да се осъществи оцветяването на клетка B6 в един от трите цвята в зависимост от въведеното в нея число, може да се използва един от следните варианти – да се заложат желаните нюанси за трите цвята в избрани клетки (в примера E2, E3 и E4) и съответният цвят да се извлече от тях; или да се присвои директно желаният цвят, като се използва функцията RGB с въведени стойности от 0 до 255 за съответните аргументи за трите основни цвята.

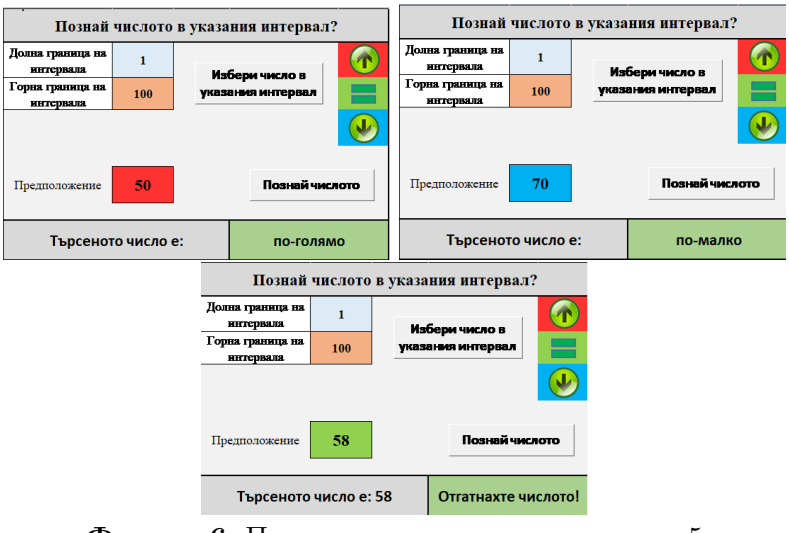
• за край, към текущия въведен в клетка A8 текст (Търсеното число e:) ще се добави (преобразуваното в низ) намерено число.

Листинг 13. Реализиране на дейността по отгатване на числото

```
Sub poznai_chisloto()
  Dim n As Integer
  n = Range("B6").Value
  If n < x Then
    Range("B6").Interior.Color = Range("E2").Interior.Color
    Range("D8").Value = "по-голямо"
  ElseIf n > x Then
    Range("B6").Interior.Color = Range("E4").Interior.Color
    Range("D8").Value = "по-малко"
  Else
    Range("B6").Interior.Color = Range("E3").Interior.Color
    Range("D8").Value = "Отгатнахте числото!"
    Range("A8").Value = Range("A8").Text + Str(n)
  End If
End Sub
```

Търсенето на елемент на редица от числа, отговарящ на определено условие, е доста често срещана задача в програмирането. Когато елементите в дадена редица са предварително сортирани, може да се приложи

и по-бърз алгоритъм за търсене на числото, който се нарича *двоично търсене*. В случая, ако сме задали интервал [1,100] и имаме сортирана редица, алгоритъмът за двоично търсене се основава на следното: прави се проверка дали търсеното число се намира по средата на редицата. Ако търсеното число не попада там, но е по-малко от средния елемент, тогава търсенето продължава в лявата половина на редицата, в противен случай — в дясната половина. Дейността се повтаря, като на всяка следваща стъпка се скъсява наполовина интервалът, в който се търси числото.



Фигура 6. Примерно изпълнение на задача 5

4. Заключение

Предложеният подход и учебни задачи дават възможност учениците да развият и задълбочат своите алгоритмични знания и умения, като ги приложат в нов контекст и среда за програмиране чрез използване на вградените средства за програмиране на потребителските продукти с общо предназначение. Същевременно предложеният подход способства и за развитие на практическите и приложните умения на учениците в областта на компетентно използване на информационните технологии[2].

Работата ще е полезна за учители, преподаващи информатика и информационни технологии в гимназиален етап.

БЕЛЕЖКИ

1. Министерство на образованието и науката. *Учебна програма по информатика за 8. клас (общообразователна подготовка).* https://www.mon.bg/nfs/2018/01/up_8kl_informatika_zp.pdf
2. Министерство на образованието и науката. *Учебна програма по информационни технологии за 10. клас (общообразователна подготовка).* https://www.mon.bg/nfs/2018/01/pril3_up_10kl_it.pdf

REFERENCES

ANEVA, S., TODOROVA, E., 2024. Development of digital and algorithmic skills for students using Visual Basic for Applications. *Mathematics and Informatics*, vol. 67, no. 3, 2024, pp. 314 – 335 (in Bulgarian).

ANEVA, S., TODOROVA, E., 2021. Prospects for Developing Students' Algorithm Skills in Teaching the Subject "Computer Modeling and Information Technologies" in Middle School, *Anniversary International Scientific Conference "Computer Technologies and Applications"*, pp. 37 – 46 (in Bulgarian).

GROZDEV, S., GAROV, K., 2008. On the system of supportive problems in the preparation for participation in Informatics Olympiads (Combinatorial objects and algorithms). *Proceedings of Thirty Seventh Spring Conference of the Union of Bulgarian Mathematicians*, pp. 304 – 311 (in Bulgarian).

GAROV, K., 2010. Zadachite v obuchenieto po informatika i informatsionni tehnologii, *Natsionalna konferentsia „Obrazovanieto v informatsionnoto obshtestvo"*, pp. 95 – 101 (in Bulgarian).

GAROV, K., 2013. *Nyakoi metodicheski aspekti na obuchenieto po informatika i informatsionni tehnologii,* Paisii Hilendarski, Plovdiv (in Bulgarian).

DUREVA, D., 2003. *Problemi na metodikata na obuchenie po informatika i informatsionni tehnologii*, Neofit Rilski, Blagoevgrad (in Bulgarian).

MANEV, K., MANEVA, N., HRISTOVA, V., 2017. *Informatics for 8th grade*, Izkustva, Sofia (in Bulgarian).

RAHNEV, A., GAROV, K., 1988. Programirane na rekurentni formuli. *Matematika*, vol. 4, pp. 34 – 39 (in Bulgarian).

RAHNEV, A., GAROV, K., 1988a. Nyakoi zadachi po programirane, svarzani s chislata na Fibonachi. *Matematika*, vol. 8, pp. 35 – 37 (in Bulgarian).

RAHNEV, A., GAROV, K., GAVRAILOV, O., 1990. *Beysik v primeri i zadachi,* Narodna prosveta, Sofia (in Bulgarian).

TODOROVA, E., ANEVA S., CHILIKOVA S., DELCHEVA P., 2021. Forming and Developing Cognitive Skills in Teaching "Computer Modeling and Information Technologies" in Middle School. *Anniversary International Scientific Conference "Computer Technologies and Applications"*, pp. 103 – 113 (in Bulgarian).

ЛИТЕРАТУРА

АНЕВА, С., ТОДОРОВА, Е., 2024. Развитие на дигитални и алгоритмични умения на учениците чрез използване на Visual Basic for Applications. *Математика и информатика*, том 67, № 3, с. 314 – 335.

АНЕВА, С., ТОДОРОВА, Е., 2021. Възможности за развитие на алгоритмични умения на учениците в обучението по предмета „Компютърно моделиране и информационни технологии" в прогимназията. *Юбилейна международна конференция „Компютърни технологии и приложения"*, с. 37 – 46.

ГРОЗДЕВ, С., ГЪРОВ, К., 2008. За системите от опорни задачи при подготовката за участие в олимпиади по информатика (Комбинаторни обекти и алгоритми). *Тридесет и седма пролетна конференция на Съюза на математиците в България*, с. 304 – 311.

ГЪРОВ, К., 2010. Задачите в обучението по информатика и информационни технологии. *Национална конференция „Образованието в информационното общество"*, с. 95 – 101.

ГЪРОВ, К., 2013. *Някои методически аспекти на обучението по информатика и информационни технологии*, УИ „Паисий Хилендарски", Пловдив.

ДУРЕВА, Д., 2003. *Проблеми от методиката на обучение по информатика и информационни технологии*, УИ „Неофит Рилски", Благоевград.

МАНЕВ, К., МАНЕВА, Н., ХРИСТОВА, В., 2017. *Информатика за 8. клас общообразователна подготовка*, Изкуства, София.

РАХНЕВ, А., ГЪРОВ, К., 1988. Програмиране на рекурентни формули. *Математика*, № 4, с. 34 – 39.

РАХНЕВ, А., ГЪРОВ, К., 1988а. Някои задачи по програмиране, свързани с числата на Фибоначи. *Математика*, № 8, с. 35 – 37.

РАХНЕВ, А., ГЪРОВ, К., ГАВРАИЛОВ, О., 1990. *Бейсик в примери и задачи*, Народна просвета, София.

ТОДОРОВА, Е., АНЕВА С., ЧИЛИКОВА С., ДЕЛЧЕВА П., 2021. Формиране и развитие на познавателни умения в обучението по „Компютърно моделиране и информационни технологии" в про-

гимназията. *Юбилейна международна конференция „Компютърни технологии и приложения“*, с. 103 – 113.

USING VISUAL BASIC FOR EXCEL APPLICATIONS FOR DEVELOPING STUDENTS' PROGRAMMING SKILLS

Abstract. The paper presents an approach for developing and refining the students' practical and applied skills in programming by using the capabilities of Visual Basic for Excel Applications. Five problems are given along with their solutions and are realized through integrated features for programming in Excel. These examples help facilitate the execution of the following activities – determining the number of even and odd numbers in a finite set of elements and extracting the corresponding lists; finding all prime numbers in a given interval; merging the elements of two ordered one-dimension arrays with a determined number of elements into a third ordered array; finding a randomly generated integer within a given interval. The suggested approach and problems aim to help students apply the acquired knowledge and skills having to do with using branched and cyclical algorithmic constructions, working with one-dimension arrays as well as creatively using the acquired digital skills for processing spreadsheets data in the area of programming in the high school stage.

Keywords: informatics; information technologies; algorithmic skills; algorithms; Visual Basic for Applications; spreadsheets; digital skills; digital creativity

✉ **Dr. Stefka Aneva, Assoc. Prof.**
ORCID iD: 0000-0002-0552-3412
Faculty of Mathematics and Informatics
University of Plovdiv „Paisii Hilendarski“
Plovdiv, Bulgaria
E-mail: stfaneva@uni-plovdiv.bg

✉ **Dr. Elena Todorova, Assist. Prof.**
ORCID iD: 0000-0002-1375-4106
Faculty of Mathematics and Informatics
University of Plovdiv „Paisii Hilendarski“
Plovdiv, Bulgaria
E-mail: etodorova@uni-plovdiv.bg