Пловдивски университет „Паисий Хилендарски”
Факултет по математика и информатика
Катедра „Компютърна информатика”

Галя Илиева Шивачева

ВИРТУАЛНА ЛАБОРАТОРИЯ ЗА ОБУЧЕНИЕ ПО ПРОГРАМИРАНЕ

АВТОРЕФЕРАТ

на дисертационен труд
за присъждане на образователна и научна степен „Доктор“
по област на висше образование
4. Природни науки, математика и информатика
професионално направление 4.6 Информатика и компютърни науки
докторска програма „Информатика“

Научен ръководител:
проф. д.м.н. Георги Атанасов Тотков

Рецензенти:
проф. д-р Маргарита Стефанова Теодосиева
доц. д-р Валентина Николаева Войноховска

Пловдив
2018 г.

Дисертационният труд е обсъден и насочен за защита пред научно жури, на заседание на катедра „Компютърна информатика” при Факултета по математика и информатика на Пловдивския университет „Паисий Хилендарски” на 28.06.2018 г.

Дисертационният труд „*Виртуална лаборатория за обучение по програмиране* ” съдържа 126 страници. Списъкът на използваната литература включва 91 източника (от които 32 на български, 13 на руски, 28 на английски и 18 Интернет-източника на английски, български и руски на латиница). Списъкът на авторските публикации по темата се състои от 10 заглавия.

Материалите по защитата са на разположение на интересуващите се в Деканата на Факултета по математика и информатика, Нова сграда на ПУ „Паисий Хилендарски“ (бул. „България” № 236, Пловдив), всеки работен ден от 8:30 до 17 часа.

Защитата на дисертационния труд ще се състои на 05.10.2018 г. от 12:30 ч. в Заседателната зала на ПУ „П. Хилендарски“ (Нова сграда, бул. „България“ № 236).

Автор: Галя Илиева Шивачева
Заглавие: „Виртуална лаборатория за обучение по програмиране”

Abstract

VIRTUAL LABORATORY FOR TEACHING PROGRAMMING

Galia Ilieva Shivacheva

The main objective of the PhD thesis is to design and implement a Virtual Laboratory for Teaching Programming (VLTP), which:

- provides opportunities for improving the quality of the learning process;
- helps self-preparation;
- provides conditions for distance learning.

A "virtual lab" is reviewed and a definition is formulated. Classifications of virtual laboratories from different authors are presented and a new one is defined. The advantages and disadvantages of existing laboratories from the point of view of their implementation and application in the field of programming are outlined. A definition of a virtual training programming laboratory is formulated. An example is a virtual programming laboratory structure. Various software for building virtual laboratories is presented. Tools for visualization in programming training are analyzed.

The main stages in solving a programming task are formulated. We have accumulated frame models (AFMs) and frames-prototypes have been created to declare variables and describe programmatic invariants suitable for automating programming training. Existing virtual labs and other programming training tools have been analyzed. VLTP is designed: the functional requirements are defined, the conceptual, functional and architectural model and the databases used are defined.

A VLTP environment is presented as a structure and functional description. It proposes a methodology for its use for lectures, exercises, self-preparation and verification and control of accumulated knowledge and skills. There are types of tasks that can be present in the programming training, and to use them to implement them.

Methodological training is being studied along with the opportunities for using ICT. A "reading with comprehension" programming training is developed - understanding the program code by detecting sample frames of the respective programming invariants. Four experiments for programming training using AFM and VLTP were presented at Faculty of Technics and technologies, Yambol, Trakia University.

The following development perspectives can be identified:

- Automatic generation of program code based on verbal description with invariants;
- Design of systems of type AFM (for different topics, didactic units, etc.);
- Research of algorithms and methods for extracting and aggregating data from learning materials in e-learning processes "tagged" with AFM;
- Development of methodologies for the use of AFM in e-learning.

Съдържание

Използвани съкращения

АФМ	–	Акумулативни фреймови модели
БД	–	База от данни
ВЛ	–	Виртуална лаборатория
ВЛОП	–	Виртуална лаборатория за обучение по програмиране
ЕМдО	–	Евристичен метод на обучение
ЕП	–	Език за програмиране
ИКТ	–	Информационни и комуникационни технологии
ИУЕ	–	Интерактивна учебна единица
МдМО	–	Метод на моделното обучение
МдПгО	–	Метод на програмираното обучение
МдПрО	–	Проблемен метод на обучение
ОИМд	–	Обяснително-илюстративен метод
СОП	–	специални образователни потребности
ТрЕУ	–	Тракийски електронен университет, гр.Стара Загора
ТрУ	–	Тракийски университет, гр.Стара Загора
ФТТ	–	Факултет “Техника и технологии“, гр.Ямбол
UOC	–	Open University of Catalonia
VLTP	–	Virtual Laboratory for Teaching Programming
VPL	–	Virtual Programming Lab(модул в Moodle)

Увод

В стратегията Европа 2020 на Европейския съюз една от петте основни цели е свързана с образованието и неговият принос за интелигентния растеж на обществото. В България има създадени различни програми, свързани с подобряване на процеса на обучение и повишаване на неговата ефективност. Голяма част от тях са свързани с използване на новите информационни и компютърни технологии. Един от етапите на Националната програма за създаване на **Виртуално Образователно Пространство** е създаването на **виртуални лаборатории** [Смрикаров, 2009]. Те са пряко свързани с **електронното обучение** като от една страна могат да бъдат негов елемент, т.е. част от него, а от друга могат да използват за реализацията си някои от средите за електронно обучение. Повечето от предимствата на електронното обучение се превръщат автоматично и в предимства на виртуалните лаборатории. Съвременните тенденции в електронното обучение [Тотков и др., 2014г] и методиката на този тип обучение [Тотков и др., 2014в] са почти напълно валидни и за виртуалните лаборатории.

Освен функциите за писане, редактиране, съхраняване, тестване и изпълнение на програмен код [VPL], които са реализирани във VPL една виртуална лаборатория за обучение по програмиране (ВЛОП) е добре да предоставя и други възможности като например: визуализиране на програмен код, представяне на понятия, оператори, алгоритми и структури от данни чрез анимация, визуализация или симулация и решаване на задачи по програмиране, свързани с фреймови модели.

Структура на дисертацията

Дисертационният труд се състои от Списък на използваните съкращения, Списък на фигурите, Списък на таблиците, Увод, 4 (четири) глави (представящи решения и резултати на поставените задачи), Заключение, Списък на използваната литература, Приложения, Списък на авторските публикации по темата, Списък на забелязани цитирания на публикации по темата и Декларация за оригиналност на резултати и приноси.

Основният текст на дисертационния труд се състои от 126 (сто двадесет и 6) страници и се съпровожда от две приложения (шест страници), които съдържат таблици, свързани с първи и трети експеримент.

Цел и задачи

Цел на настоящия дисертационен труд е **проектиране и реализиране** на ВЛОП, която предоставя възможности за повишаване на качеството на учебния процес; подпомага самостоятелната подготовка и сигурява условия за дистанционно обучение.

За постигането на поставената цел, е необходимо да бъдат решени следните конкретни **задачи**:

Задача 1. Да се направи **анализ на състоянието** на изследванията в областта на ВЛ и по-конкретно на ВЛОП.

Задача 2. Да се определят **функционалните изисквания към среда от тип „ВЛОП“** и да се проектират концептуален и функционален модел, архитектура и бази от данни.

Задача 3. Да се реализира **прототип на среда за ВЛОП**.

Задача 4. Да се организират и проведат **експерименти** за виртуално обучение по програмиране.

В **Глава 1. Състояние на изследванията** е направен преглед на понятията „виртуална лаборатория“ (ВЛ) и ВЛОП. Представени са класификации на ВЛ. Анализирани са средства за визуализация в обучението по програмиране. Дефинирани са целта и задачите на дисертационния труд.

В **Глава 2. Модел и архитектура на ВЛОП** са формулирани основните етапи при решаване на задача по програмиране. Представени са акумулативни фреймови модели. Формулирани са функционалните изисквания и са описани концептуалния , функционалния и архитектурния модел на ВЛОП.

В **Глава 3. Реализация на среда от тип ВЛОП** е представена среда от тип ВЛОП като структура и функционално описание. Предложена е методика за използването ѝ за лекции, упражнения, самоподготовка и проверка и контрол на натрупаните знания и умения. Представени са видове задачи, които могат да присъстват в обучението по програмиране и за реализирането им да се използва ВЛОП.

В **Глава 4. Виртуално обучение по програмиране** от методическа гледна точка е разгледано обучението по програмиране заедно с възможностите за използване на ИКТ в него. Представени са четири експеримента за обучение по програмиране с използване на АФМ и ВЛОП.

В **Заключението** получените резултати са обобщени и систематизирани, като са посочени основните научни, научно-приложни и приложни приноси на дисертационния труд.

Формулирани са перспективи за бъдещо развитие на дисертационната тематика.

Апробация на резултатите

Основните резултати на изследването са докладвани на катедрени и докторантски семинари, на национални и международни научни форуми.

Резултатите от дисертационното изследване са представени в 10 (десет) публикации:

– 1 (една) в списание и 9 (девет) – в трудовете на конференции (8 международни и 1 национална)

– 5 (пет) на български език, 4 (четири) на английски език и 1 (една) на руски език.

За сравнение, в следващата таблица са дадени изискуемия минимален брой точки по групи показатели на *Националните изисквания за присъждане на ОНС „доктор“ в ПН 4.6. Информатика и компютърни науки* **[Постановление №122 на МС, 2018]**[1] и на постигнатия брой точки според материалите, съпровождащи процедурата (дисертационен труд, публикации, цитирания на публикации, участия в научни форуми и проекти и др.).

Група от показатели	**Съдържание**	**Брой точки**	
		Изискуем	**Постигнат**
А	Показател 1	**50**	**50**
Г	Сума от показателите от 5 до 9	**30**	**109,99**
Д	Сума от показателите от 10 до 12	**-**	**15**
	Общо:	**80**	**174,99**

[1] Прието и публикувано след стартиране на процедурата по защита.

Благодарности

Изказвам своите най-искрени благодарности на моят научен ръководител – проф. д.м.н. Георги Тотков за придобитите знания и умения, проявената отзивчивост и огромна подкрепа по време на целия период на работа върху дисертацията!

Благодарна съм и на колегите от катедра „Компютърна информатика“ на ПУ, по-специално на доц. Сомова и на колегите от катедра „ЕЕА“ на ФТТ – Ямбол, по-специално на доц.Недева за оказаната подкрепа и съдействие при провеждане на дисертационните експерименти.

Признателна съм и за финансовата подкрепа на изследването, оказана по проект №2 ФТТ/30.04.2015г. „Приложение на виртуални лаборатории във висшите училища“, Факултет „Техника и технологии“ - Ямбол, Тракийски университет – Стара Загора и проект МУ17-ФФ-023 „Акумулативни фреймови модели за извличане и агрегиране на данни за знания и процеси в обучението“ към Фондове „Научни изследвания“ на ПУ „Паисий Хилендарски“.

Глава 1. Състояние на изследванията

1.1 За понятието „виртуална лаборатория“

1.1.1. Що е виртуална лаборатория?

На базата на направения аналитичен обзор е формулирано следното определение:

ВЛ [Шивачева и др., 2015] е **компютърно базирана лаборатория, в която интерактивно чрез софтуерна симулация на реални обекти, устройства или процеси или чрез отдалечен достъп до реална лаборатория се осигурява възможност за провеждане на експерименти и извършване на различни измервания.**

В първия случай са **виртуални елементите** на лабораторията, а във втория случай е **виртуално присъствието** на потребителите. Тези лаборатории са подходящи за използване от учащи се, особено **дистанционна форма** на обучение и от изследователи, работещи по **общ проект** и разположени в **различни географски точки** [Шивачева и др., 2015].

В процеса на работа по проучването на съществуващи ВЛ и създаването на прототип на ВЛОП е формулирано и определението:

ВЛ е **компютърно базирано средство за софтуерно моделиране и симулиране (вкл. планиране, осъществяване и оценяване) на реални процеси чрез виртуализация и/или отдалечен достъп до обекти и методи за тяхната обработка.**

1.1.2. Класификация на виртуалните лаборатории

На базата на разгледаните класификации на виртуални лаборатории може да се формулира нова класификация, която се състои от следните видове лаборатории:

- Лаборатории **с отдалечен достъп,** които използват **реално оборудване;**
- Лаборатории, **базирани на симулационно моделиране на процеси и явления;**
- Лаборатории, **базирани на инструменти за измерване;**
- **Хибридни** лаборатории**, които комбинират два или три** от изброените **видове лаборатории** в тази класификация [Шивачева и др., 2015].

1.2 Виртуални лаборатории в обучението по програмиране

Програмирането е част от науката информатика. Състои се от поредица от абстрактни понятия, за разбирането на които обикновено не е достатъчно само словесно описание, а са необходими и примери от реални обекти и/или някаква визуализация.

Развитието на алгоритмичното мислене у обучаваните е бавен и труден процес. Това е различен тип мислене, което се изгражда с времето на базата на натрупване на опит. Затова за подпомагането на този процес е необходимо да се създаде ВЛ, която да намали абстракцията и да улесни и ускори процеса на развитие на алгоритмичното мислене.

Във ВЛОП могат да се симулират основни понятия, оператори от езиците за програмиране, базови алгоритми и структури от данни с цел по-добро разбиране, практическо усвояване и прилагане.

1.2.1. Обща структура за ВЛ по програмиране

Основните ресурси на ВЛ по програмиране [Prieto-Blazquez, 2009] са**: технологични, педагогически и стратегически и ресурси на академичния състав.**

Технологичните ресурси се състоят от: **виртуална комуникационна среда, симулатор, отдалечена лаборатория, виртуална машина** и **инструмент за автоматична оценка**.

Виртуалната машина е инструмент, който позволява на потребителите да създават отделни среди, всеки от които емулира хардуера на физически компютър.

Инструментът за автоматична оценка позволява да се проведат интерактивни процеси между студентите и автоматичната система за оценка, която дава възможност на учениците да научат повече за процесите на итерация. **Педагогическите и стратегически** ресурси включват: **методика на обучение, съпровождаща документация и други материали** и **оценка**.

Ресурсите **на академичния състав** включват **преподавател**.

"Виртуалните **преподаватели"** във ВЛ по програмиране са членове на академичните среди, които помагат на студентите да постигнат индивидуалните си цели, като всеки преподавател има точно определени задължения [Prieto-Blázquez, 2009]**.**

1.2.2. Средства за визуализация в обучението по програмиране

За намаляване на проблемите в обучението, свързани със синтаксиса на езиците за програмиране и за по-лесното възприемане на алгоритмите при изучаването им се използват средства за визуализация, които онагледяват и понижават нивото на абстрактност.

Създадени са средства за визуализация, които реализират една или повече от следните функции:

- Описание на алгоритъм чрез **блок-схемен език**;
- Описание на алгоритъм чрез **език за програмиране**, елементите на който са представени чрез отделни **блокове** и създаването на програмата наподобява подреждане на пъзел;
- **Тестване** на алгоритъм, представен чрез блок-схема или чрез език за програмиране;

- Изпълнение стъпка по стъпка на програмен код и интерактивно представяне в табличен вид на имената на променливите и текущите им стойности по време на изпълнението.

Flow Chart Visual Programming Language

FlowProgramming.exe [FlowProgramming] е приложна програма за описание и тестване на алгоритми, описани чрез блок-схема.

Snap

Snap! [Snap] е **визуален език за програмиране с „плъзгане и пускане“**. Представлява своеобразно разширение на **Scratch**(http://scratch.mit.edu) с добавена възможност за създаване на собствени блокове, представящи визуално като елемент на пъзел оператор или друг компонет от език за програмиране.

Snap! дава възможност за **генериране и тестване на програми** чрез използване на **готови блокове**.

Flowgorithm

Flowgorithm [Flowgorithm] е **безплатно** приложение, което позволява създаване на програма по блок-схема. Flowgorithm има следните **характеристики** и **предимства**:

- Графичен прозорец за използваните променливи (представени с име, стойност и цвят, определящ типа им);
- Интерактивно генериране на програмен код на 18 езика за програмиране;
- Възможността за **изпълнение стъпка по стъпка** и **извеждане на имената и текущите стойности на променливите**, както и на **имената на** използваните **функции**.

Python Tutor

Онлайн **Python Tutor** е **уеб базиран графичен инструмент за визуализация на изпълнението на програми**, написани на осем езици за програмиране [Шивачева, 2018].

1.3 Изводи

Предимствата на приложението на ВЛ в обучението по програмиране са съществени. Те дават възможност за доближаване на представянето на учебния материал до вижданията на самите обучаеми, до техния стил на живот, изцяло свързан и зависим от съвременните технологии. Осигуряват условия за безопасна работа по всяко време и от всяко място, за многократно изпълнение на едни и същи експерименти или симулации. Създават условия за електронно и дистанционно обучение и за обучение на хора със СОП (специални образователни потребности). ВЛОП повишават интереса и намаляват равнището на абстрактност, което способства за по-бързо и по-лесно възприемане на учебния материал и за постигането на по-високи резултати. Подходящи са за обучение на хора от новото поколение, което постоянно е в досег с ИКТ.

ВЛОП е компютърно базирано приложение за интерактивни симулации на алгоритми и структури от данни.

Предполага: моделиране и симулиране на алгоритми и на използваните от тях структури от данни.

Предимства: по-лесно възприемане от обучаваните и развитие на тяхното алгоритмично мислене.

Глава 2. Модел и архитектура на ВЛОП

2.1. Основни етапи при решаване на задача по програмиране

Процесът на решаване на задача с използване на компютър включва следните етапи:

Е1. Постановка на проблема

На този етап **от студента, с помощта на преподавателя – като част от обучението** се построява описателен информационен модел на обекта или процеса. Събира се (ако е необходимо) информация за задачата (или подобни на нея), формулира се условието на задачата; определят се крайните цели на решението на задачата; определя се формата на извеждане на резултатите; описват се данните (вкл. техния тип, диапазон, структура и т.н.).

Е2. Построяване на формален модел на задачата

Включва изграждане на модел с характеристики, адекватни на оригинала; разглеждат се характерът и същността на величините, използвани в задачата; определя се семантиката и типа на входните, междинните и изходни данни.

Е3. Математическо или информационно моделиране

Изграждане на математически модел, вкл. математически структури от данни; математическо описание на задачата - директно или приближено; избор и обосновка на метода на решение.

Създава се математически и информационен модел като се определят идентификаторите на променливите, които ще се използват. За всяка променлива се задава описание за какво ще се използва.

Е4. Конструиране на алгоритъм за решаване на задачата, базиран на математическия модел (алгоритмизиране на задачата)

Е4.1. Избор на метод за проектиране на алгоритъма;

Е4.2. Избор на формата на записване на алгоритъм (напр. словесно);

Е4.3. Избор на методи за тестване и избор на контролни примери;

Е4.4. Проектиране и записване на алгоритъма.

Е5. Съставяне на програма (компютърно моделиране)

Е5.1. Избор на езика за програмиране;

Е5.2. Определяне на входно/изходен поток;

Е5.3. Съответствие между програмен ред и стъпка от словесното описание на алгоритъма.

Е6. Реализация на програмата за компютър (провеждане на компютърен експеримент в среда за програмиране)

Е6.1. Въвеждане / съхраняване на програмата;

Е6.2. Транслиране.

Е7. Тестване на програмата с контролни данни и отстраняване на грешки.

Е7.1. Отстраняване на синтактични грешки;

Е7.2. Отстраняване на семантични грешки;

Е7.3. Отстраняване на грешно проектирана логическа структура;

Е7.4. Тествания с контролните данни;

Е7.5. Анализ на резултатите от тестванията и добавяне на нови редици от контролни данни при необходимост; усъвършенстване и актуализиране на програмата (с итеративно повтаряне на етапи от първи до седми).

Е8. Получаване/съхраняване на работеща/изпълнима версия на програма.

Е9. Изпълнение на програмата на компютър с реалните входни данни (с цел решаване на поставената задача) и анализ на резултатите (програмистът стартира програмата и задава входните данни, изисквани от условието на задачата).

Е10. Архивиране (вкл. документиране и съхраняване) на програмата за следваща експлоатация.

2.2. Акумулативни фреймови модели

Акумулативните фреймови модели са относително самостоятелни, логически обособени единици, които се отличават с възможността за многократно използване в различни ситуации. За тяхното описание се използва именована фреймова структура, съставена от слотове. От своя страна, всеки слот е с уникално име (различно от имената на останалите слотове на АФМ) и със съдържание от точно определен тип (елементарен или съставен).

В Табл. 2.1. и Табл. 2.2. са създадени шаблони на фрейм-прототип за деклариране на променливи и на фрейм-прототип за описание на програмни инварианти. Всяка променлива от фрейма „Деклариране на променливи“ има характеристики: тип, име, описание (за какво се използва тази променлива) и програмен код, с който се реализира нейното дефиниране (запазване на памет за променливата). Всеки масив, представен чрез фрейма „Деклариране на масив“ има характеристики: тип (на елементите на масива), име, описание (за какво се използва масива и максимален брой елементи) и програмен код, с който се реализира дефинирането му (запазване на памет за масива). Всеки фрейм-инвариант се представя чрез слотове за тип, име и описание, свързани с променливите, които се използват в програмния код за реализацията му.

№	Име на фрейм	Променлива			Програмен код
		Тип	Име	Описание	
1.	Деклариране на променлива				
2.	Деклариране на масив				

Таблица 2.1. Шаблон на фрейм-прототип за деклариране на променливи (обикновени и индексирани)

№	Име на фрейма-инвариант	Слотове			Програмен код
		Име	Тип	Описание	
1.					
...	...	…	…	…	

Таблица 2.2. Шаблон на фрейм-прототип за описание на програмни инварианти

С въвеждането на АФМ, в обучението по програмиране се обособяват следните **групи задачи**:

А. Запознаване с често използвани инварианти за програмиране (спец. със съдържанието на слот „Програмен код");

Б. Анализ на непознат програмен код, откриване на инварианти (вкл. и на нови), както и на стойностите на техните параметри (съдържание на съответните слотове);

В. Модифициране на непознат програмен код (с промяна на съдържанието на слотовете на фрейми-инварианти и въвеждане на нови инварианти);

Г. Проектиране на решения на задачи за програмиране под формата на словесни описания (съставени от фрейми-инварианти с конкретно съдържание на слотовете).

Д. Самостоятелно проектиране и писане на програмен код (по дадени спецификации).

Групите от задачи от А до Д могат да се реализират чрез различни действия, свързани с табличното представяне на АФМ.

За изпълнение на задачите от група А при изучаване на типовете данни и дефиниране на променливи се попълва от преподавателя в дискусия със студентите таблица от вида на Табл. 2.1., а при изучаване на оператори и алгоритми – таблица от вида на Табл. 2.2.

За задачи от група Б при зададен програмен код студентите откриват познати фрейми в този код и попълват Табл. 2.1. и Табл. 2.2.

При задачи от група Д програмния код, получен при решаване на задачи от група В или Г го оформят като програма в съответна среда за програмиране. Компилират и тестват с подходящи входни данни [Шивачева и Тотков, 2017].

2.3. Сравнителен анализ на ВЛ и други средства за обучение по програмиране

В тази глава анализът е от гледна точка на това, кои от етапите могат да се реализират чрез съответните средства.

Snap

Към **Snap** е добавено приложението **Snap Cloud**(Snap облак), което позволява съхраняване (**десети** етап) и изтегляне на проекти онлайн. Snap! [Snap] дава възможност за генериране (**пети** етап) и тестване (**седми** и **девети** етап) на програми чрез използване на готови блокове като премахва проблемите с допускане на синтактични грешки при писане и осигурява акцент върху алгоритъма, а не върху описанието му чрез език за програмиране.

Flowgorithm

Flowgorithm [Flowgorithm] е безплатно приложение, което позволява да се опише алгоритъм чрез блок-схемен език и интерактивно генериране на програмен код на осемнадесет езика за програмиране на база на създадената блок-схема(**четвърти** и **пети** етап). Представената по този начин програма може да се изпълни директно или да се съхрани като cpp файл(**шести**, **седми**, **девети** и **десети** етап).

Moodle Virtual Programming Lab

Moodle Virtual Programming Lab [**VPL**] е модул за оценяване на програмен код и притежава следните функции:

- Писане, редактиране и съхраняване на програмен код (**пети**, **шести** и **десети** етап);
- Тестване на програма (**седми** етап);
- Изпълнение на програма в браузър (**девети** етап).

Реализиран е като плъгин на системата за електронно обучение Moodle и може да се използва за програми на повече от десет езика.

Тестването и оценяването се извършва чрез използване на скриптове,които могат да се задават за изпълнение, отстраняване на грешки или оценяване, или чрез поредица от контролни примери, записани в зададен формат в текстовия файл "**vpl_evaluate.cases**".

VPLab

В Политехническия университет на Каталуния са създадени 29 виртуални лаборатории по програмиране (**VPLab**) [Prieto-Blazquez, 2009]. Примерната им обща структура се състои от девет ресурса, като свързани с етапите, които разглеждаме са:

- **Виртуална машина** – осигурява една и съща операционна система и среда за програмиране (от **пети** до **десети** етап) за всички студенти;
- **Инструмент за автоматична оценка** (тестване с контролни примери – една част от тях са предварително известни на студентите, а другата част не са достъпни за студентите на седми етап).

В Табл. 2.3. за всеки от основните етапи със знак плюс (+) или минус (-) за съответното средство се отбелязва дали може или не се използва за реализирането му.

№	Етап	Flowgorithm	Snap	VPL	VPLab
1	Постановка на проблема	–	–	–	–
2	Формално построяване на модел на задачата	–	–	–	–
3	Математическо или информационно моделиране	–	–	–	–
4	Конструиране на алгоритъм	+	–	–	–
5	Съставяне на програма	+	+	+	+
6	Реализация на програмата за компютър	+	–	+	+
7	Отстраняване на грешки и тестване на програмата	+	+	+	+
8	Получаване на работеща (изпълнима версия на) програма	–	–	+	+
9	Изпълнение на програмата на компютър с реалните входни данни и анализ на резултатите	+	+	+	+
10	Архивиране на програмата за следваща експлоатация	+	+	+	+

Таблица 2.3. Сравнителен анализ

От сравнителният анализ се установи, че за първите три етапа, както и за някои от другите етапи като четвърти и осми няма открити съществуващи виртуални средства за реализирането им.

За реализацията на обучение по програмиране, базирано на АФМ също няма достатъчно създадени средства. За това възниква необходимостта от създаване на среда от тип ВЛОП, която да позволи решаването на групите от задачи, базирани на АФМ и преминаването през всички етапи от процеса на решаване на задача по програмиране.

2.4. Концептуален модел на ВЛОП

На фиг. 2.1. е представен концептуален модел на ВЛОП. Той е с високо ниво на абстракция и определя три основни групи ресурси: **ресурси за обучение, база от данни за потребителите на ВЛОП** и **средства за комуникация**.

Ресурсите за обучение осигуряват необходимия минимум от знания за реализирането на учебния процес за обучение по програмиране и се състоят от:

- *модули на ВЛОП* – осем на брой, обхващащи основни теми от теорията на програмирането и състоящи се от различен брой *интерактивни учебни единици* (самостоятелно обособени единици, реализиращи конкретна тема, свързана с основни понятия, оператор или алгоритъм и представени чрез теоретична част, тестови въпроси и визуализация, анимация и симулация), описани в структурата на ВЛОП;
- *тестове* – четири на брой, обхващащи въпроси от различен тип, свързани със съответните модули. Всеки тест се отнася до два модула;

- *визуализация, анимация и симулации* – обособена за целта част за съответна интерактивна учебна единица (ИУЕ), за която е подходящо такова представяне на информацията с цел намаляване на абстрактността на понятията и увеличаване на нагледността за по-лесно възприемане;
- *помощна информация* – допълнителни обяснения към симулацията, за която се отнасят и могат да се използват при необходимост от обучаемия.

Базата от данни за потребителите на ВЛОП съдържа:
- *профил на потребителя* – регистрационни данни и права на потребителите на ВЛОП: администратор, преподавател и обучаем;
- *допълнителна информация за обучаемите* – достигнато ниво на обучение по модули и резултати от тестовете.

Средствата за комуникация са:
- *за асинхронна* – електронна поща и дискусионен форум.
- *за синхронна* – онлайн чат.

2.5. Структура на ВЛОП

ВЛОП се състои от осем модула с различен брой интерактивни учебни единици във всеки от тях.

Тестовете включват въпроси от няколко типа, отнасящи се до знанията и уменията, придобити след обучение по интерактивните учебни единици от два близки по смисъл модула.

Регистрационните данни от профила на потребителя включват: име, фамилия, e-mail адрес, потребителско име и парола. Различните потребителите на ВЛОП: администратор, преподавател и обучаем ще имат различни права на достъп до ВЛОП.

Достигнатото ниво на обучение по модули ще съхранява информация за преминати модули и интерактивни учебни единици от обучаемия от първо и второ ниво на виртуалната лаборатория.

Резултатите от тестовете ще се съхраняват в база от данни.

2.6. Функционален модел на ВЛОП.

Модел на взаимодействие за потребител "**Преподавател**" е представен на фиг. 2.2.

Преподавателят се регистрира, получава права, идентифицира се и влиза във ВЛОП. Участва във форума като публикува мнение или нова тема. Преглежда допълнителна информация, електронна поща и съобщения. Работи с ресурси за обучение - добавя задача, добавя помощна информация, добавя интерактивна учебна единица (ИУЕ), създава симулация, създава програмен код на алгоритъм, подготвя примери, представя теория, добавя тест, добавя тестови въпроси. Проектира структура и стойности на вход/изход. Открива инварианти в зададен програмен код. Оценява проектирани структура и стойности на вход/изход и оценява четене с разбиране.

2.7. Архитектура на ВЛОП.

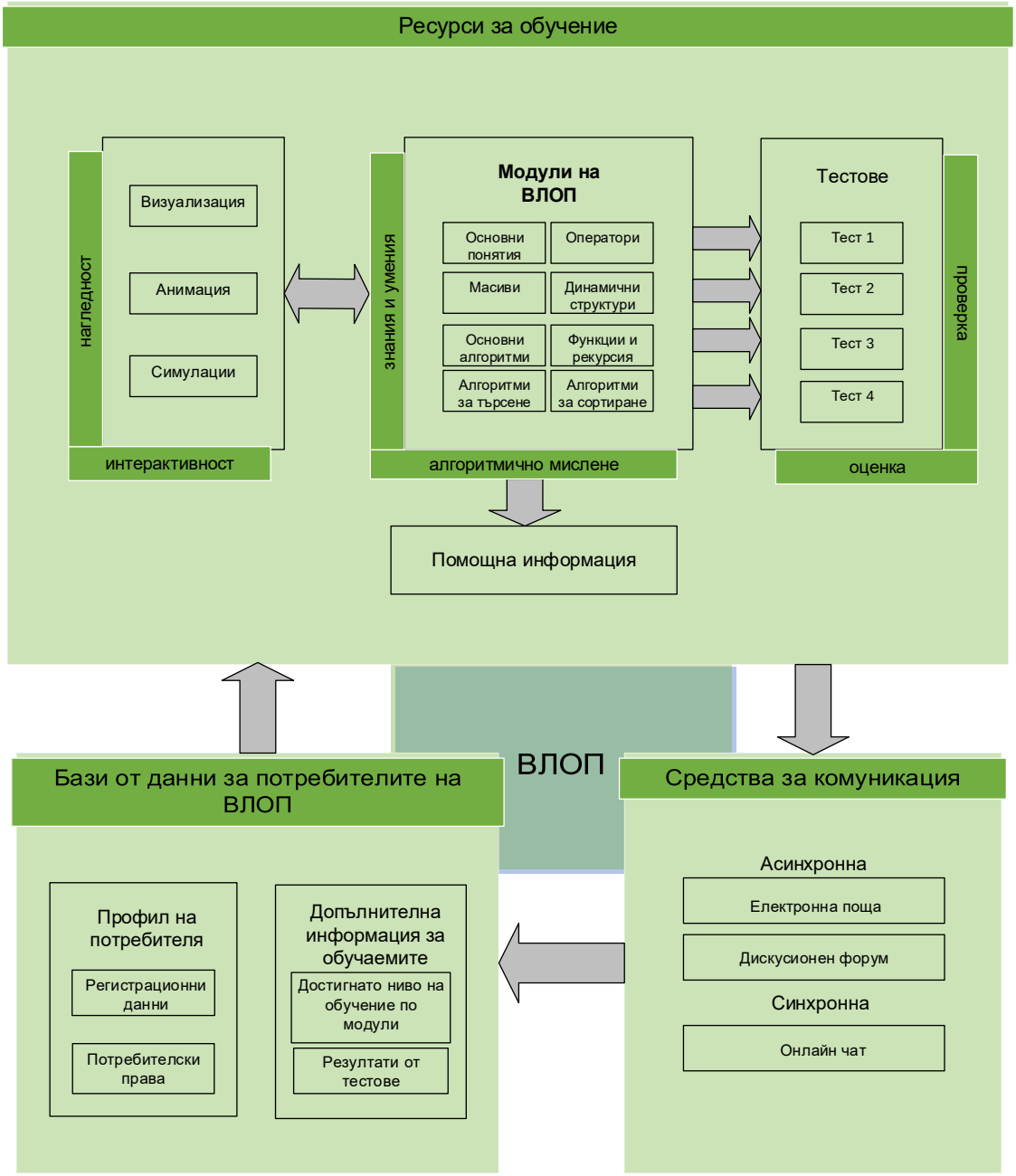
За да се осигури възможност за обучение по програмиране, базирано на АФМ за всяка от петте групи задачи трябва ВЛОП да включва функции, които ги реализират. За целта са създадени функции за: запознаване с инварианти; четене с разбиране на програмен код; модифициране на програмен код; словесно описание чрез инварианти и проектиране на вход/изход.

За прилагане на нагледните методи в обучението по програмиране са необходими средства за визуализация и функция за визуализация на алгоритми. Чрез хипервръзки (ресурс URL) се осъществява връзка към сайтове със средства

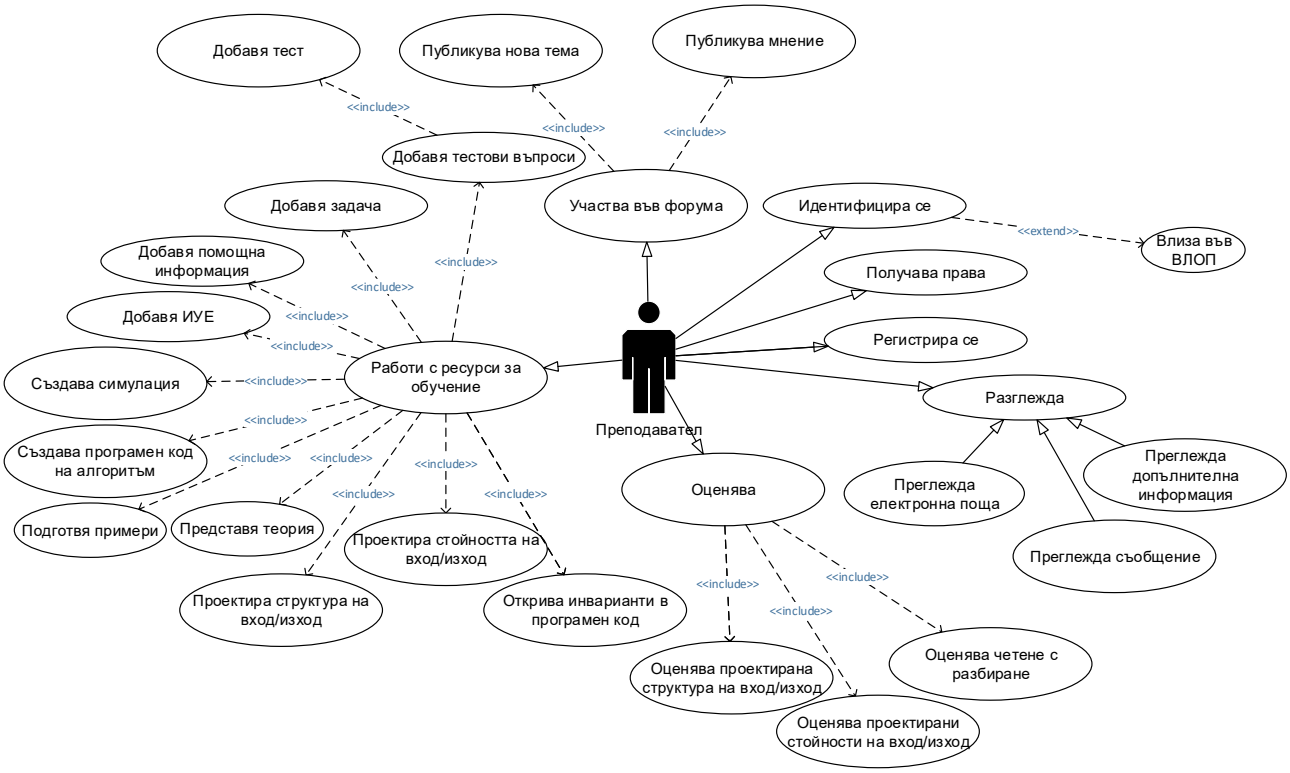
за визуализация. Във ВЛОП е реализирана функция за визуализация на алгоритми, с която са представени интерактивно алгоритмите за сортиране: метод на мехурчето, метод на пряка селекция и бързо сортиране.

За писане, редактиране, съхраняване, тестване и изпълнение на програмен код, без които процеси е немислимо обучението по програмиране е необходимо да се добави специално средство, което ги реализира. За целта модулът VPL е интегриран в Moodle, а чрез функцията „проектиране на вход/изход“ на ВЛОП се създава тестов файл за проверка и оценяване на програмен код.

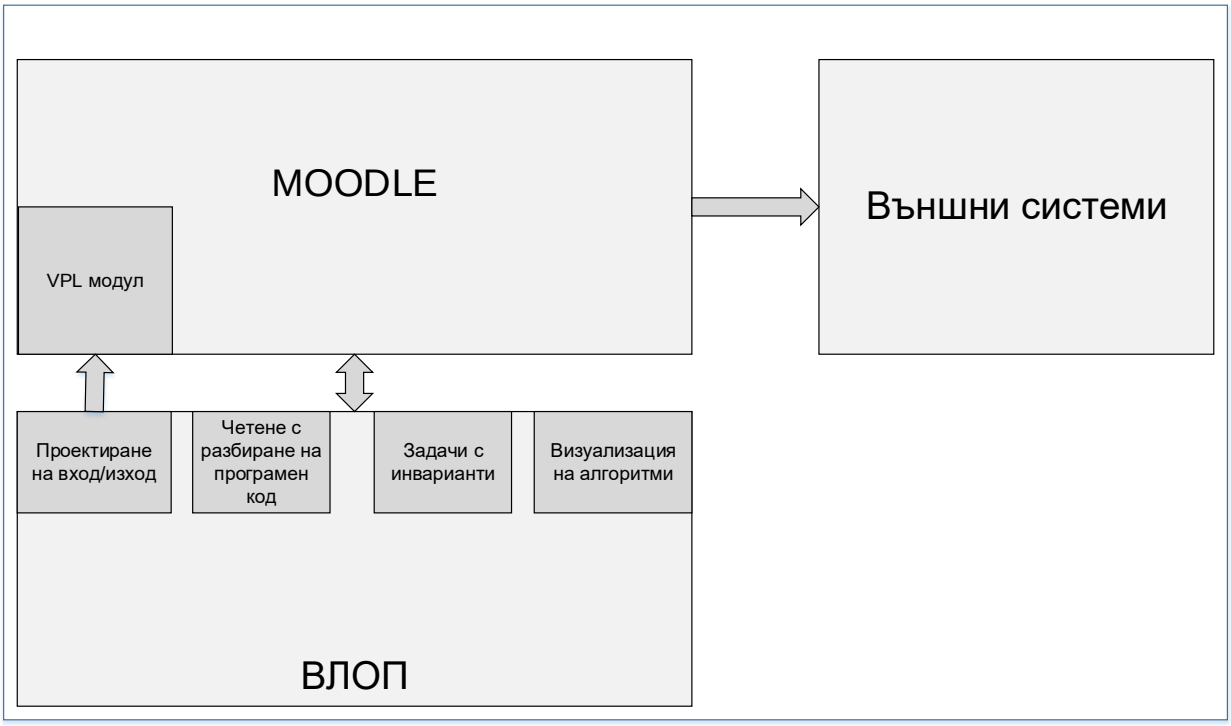
Архитектурата на ВЛОП е представена на фиг.2.3.



Фигура 2.1. Концептуален модел на ВЛОП



Фигура 2.2. Модел на взаимодействие за потребител “преподавател“



Фигура 2.3. Архитектура на ВЛОП

2.8. Бази от данни на ВЛОП.

В базите от данни се поддържа следната информация:

- потребителски профил, права за достъп (за всеки потребител);
- достигнато ниво на обучението по модули и резултати от тестовете (за обучаемите);
- ресурсите за обучение: осем модула със съответните ИУЕ; тестове за обучение по програмиране на C++ – четирите теста, включени във ВЛОП;

- помощни материали;
- БД за визуализациите, анимациите и симулациите;
- архив на БД, който се обновява автоматизирано през зададен период от време [Shivacheva et al, 2017].

Moodle е системата за електронно обучение, използвана в Тракийски университет – Стара Загора (Тракийски електронен университет) и използва базите от данни: MySQL, MariaDB, PostgreSQL, MSSQL и Oracle, в които се съхраняват данните за потребителите, ресурсите за обучение, файловете, които се използват за задания от преподавателя и за отговорите на студентите на тези задания, файлове с програмен код, създадени или записани във VPL модула и файлове за тестване на програмен код – скриптове или с тестови примери.

Във ВЛОП е използвана база от данни MySQL за създаване на таблица за съхраняване на инварианти.

2.9. Изводи.

Чрез **Flowgorithm** се реализират от четвърти до седми етап и от девети до десети (общо **шест** етапа). **VPL** модула на **Moodle** и **VPLab** могат да се приложат за етапите от пети до десети (общо **шест** етапа). Чрез **Snap** могат да се реализират пети и седми етап и от девети до десети (общо **четири** етапа).

Глава 3. Реализация на среда от тип ВЛОП

3.1. ВЛОП

В курс „Virtual Laboratory for Teaching Programming“ (VLTP) в системата за електронно обучение Moodle, използвана в Тракийски университет (Тракийски електронен университет) е инсталиран и добавен като нова дейност VPL модула на Moodle, който е създаден за автоматично тестване на програмен код. Добавени са ресурсите:

- *Страница* за визуализация на програмен код като е реализирано вграждане чрез уеб базирания графичен инструмент за визуализация на изпълнението на програми *Python Tutor* (подробно описан в първа глава);
- *URL* към сайт https://snap.berkeley.edu/snapsource/snap.html за *Snap*;
- *URL* към сайт http://vltp.atwebpages.com/, реализиращ *ВЛОП*.

ВЛОП реализира следните *функции:*

- Проектиране на структурата на входните и изходни данни и съхраняването ѝ в текстов файл;
- Проектиране на конкретни стойности на входните и изходни данни и автоматично генериране на файла vpl_evaluate.cases(текстов файл във формат за използване в модула **VPL** на Moodle);
- Откриване на инварианти в зададен програмен код и съхраняване във файл на диапазона от редове, зададен с номерата на началния и крайния ред и номенклатурния номер на инварианта, намиращ се в този диапазон от програмния код. Инвариантите се избират от предварително зададен списък с имената им;
- Откриване на инварианти в зададен програмен код и съхраняване във файл на диапазона от редове, зададен с номерата на началния и крайния ред и номенклатурния номер на инварианта, намиращ се в този диапазон от програмния код. Допълнително се задават и стойностите за слотовете на инвариантите. Инвариантите отново се избират от предварително зададен списък с имената им и след това за избрания инвариант се попълват

стойностите на полетата: **име на променлива, тип на променлива, описание и програмен код**, който го реализира;
- Запознаване с инварианти - при избор на инвариант от предварително зададен списък се извлича пълната информация за него, включително и програмен код;
- Модифициране на програмен код - избира се инвариант от предварително зададен списък и се попълват стойностите на полетата: име на променлива, тип на променлива, описание и програмен код за новата задача. При избор на инварианта „Деклариране на масив“ в полето описание се задава и максималният брой елементи на масива;
- Словесно описание чрез инварианти - Избира се инвариант от предварително зададен списък и се попълват стойностите на полетата: **име на променлива, тип на променлива, описание и програмен код** за задача, зададена с условие и се генерира текстов файл със словесното ѝ описание от зададените стойности;
- На база на текстовия файл със създадения от студента математически модел и аналогичен, създаден от преподавателя, се извършва автоматично оценяване на модела;
- На база на текстовия файл със структурата на входните и изходни данни от студента и аналогичен, създаден от преподавателя, се извършва автоматично оценяване на проектираната структура на данните;
- На база на текстовия файл със стойностите на входните и изходни данни (vpl_evaluate.cases), зададени от преподавателя чрез **VPL** модула на Moodle (инсталиран в ТрЕУ), се извършва автоматично оценяване на програмния код;
- На база на текстовия файл с диапазона от редове и номенклатурния номер на инвариант от студента и аналогичен, създаден от преподавателя, се извършва автоматично оценяване на четенето с разбиране на програмен код;
- Визуализиране на алгоритми за сортиране: метод на мехурчето, метод на пряка селекция и бързо сортиране.

В електронния курс допълнително за конкретна задача се задават следните ресурси и дейности:
- Файл с условие на задачата;
- Задания с възможност за записване на файлове от студента поотделно за всеки от първи до пети етап.

За първите три и за част от четвъртото задание студентът използва програма за текстообработка за създаване на файловете.

За четвъртото задание за избор на контролни примери може да генерира файлове чрез сайта, зададен с URL ресурса, а за проектиране и записване на алгоритъма може да се използва **Flowgorithm**.

От пети до десети етап се реализират чрез интегрирания в курса модул **VPL**.

Добавят се и задания и файлове, свързани с функциите „проектиране на вход/изход“, „четене с разбиране на програмен код“, „модифициране на програмен код“ и „словесно описание чрез инварианти“.

3.1.1. Структура на VLTP

Структурата е представена на фиг.3.1.

Проектиране на вход/изход	Четене с разбиране на програмен код	Задачи с инварианти	Визуализации на алгоритми
Проектиране	Четене с разбиране	Запознаване с инварианти	Метод на мехурчето
Оценяване	Разширено четене с разбиране	Модифициране на програмен код	Метод на пряка селекция
	Оценяване	Словесно описание чрез инварианти	Бързо сортиране

Фигура 3.1. Структура на VLTP

3.1.2. Функционално описание

Проектиране на входните и изходни данни

Списъкът за избор на вид на входните данни е представен в Табл.3.1.

Означение	Описание
int	цяло число
double	реално число
char	символ
string	текстов низ
list int	списък от цели числа
list double	списък от реални числа
list string	списък от текстови низове
list	списък
n & n int	**n** – брой на числата и наредена **n** -торка от цели числа (едномерен масив с **n** елемента)
n & n double	**n** – брой на числата и наредена **n** -торка от реални числа (едномерен масив с **n** елемента)
n & n int & k	**n** – брой на числата и наредена **n** -торка от цели числа (едномерен масив с **n** елемента) и число **к**
n & n double & k	**n** – брой на числата и наредена **n** -торка от реални числа (едномерен масив с **n** елемента) и число **к**
n & nxn int	**n** – брой на редове и брой на колони и двумерен масив от цели числа с **n** реда и **n** колони
n & nxn double	**n** – брой на редове и брой на колони и двумерен масив от реални числа с **n** реда и **n** колони
n & m & nxm int	**n** – брой на редове и **m** – брой на колони и двумерен масив от цели числа с **n** реда и **m** колони
n & m & nxm double	**n** – брой на редове и **m** – брой на колони и двумерен масив от реални числа с **n** реда и **m** колони
n & nxn int & k	**n** – брой на редове и брой на колони и двумерен масив от цели числа с **n** реда и **n** колони и число **к**
n & nxn double & k	**n** – брой на редове и брой на колони и двумерен масив от реални числа с **n** реда и **n** колони и число **к**
n & m & nxm int & k	**n** – брой на редове и **m** – брой на колони и двумерен масив от цели числа с **n** реда и **m** колони и число **к**

n & m & nxm double & k	**n** – брой на редове и **m** – брой на колони и двумерен масив от реални числа с **n** реда и **m** колони и число **к**

Таблица 3.1. Списък за избор на вид на входните данни

Проектиране

Студентът задава тестови входни и изходни данни като изпълнява следните действия:

1. От падащ списък избира вида на входните данни.
2. Въвежда конкретните стойности за всяка от тях.
3. От падащ списък избира вида на изходните данни, съответстващи на избраните входни.
4. Въвежда конкретните стойности за всяка от тях.
5. Натиска бутон „+“ за добавяне на нов тестов пример.

Създават се два текстови файла (първият с вида на входните и изходни данни за всеки тестов пример, съхранени като номер по ред на елементите от съответните списъци, а вторият с конкретните стойности на входно-изходните данни от тестовите примери във формат, подходящ за тестване с **VPL** модула на Moodle).

Файловете могат да се изтеглят на компютъра чрез натискане на бутони - един бутон за изтегляне на първия файл и втори бутон за изтегляне на другия. (Първият файл е с име inoutt.txt, а вторият - vpl_evaluate.cases).

За генериране на втория файл се използва следния програмен фрагмент:

```
document.getElementById("dwn-btn2").addEventListener("click",
function(){
     var text2="";
     var filename2 = "vpl_evaluate.cases";
     var elements3 = document.getElementsByClassName("in");
     var elements4 = document.getElementsByClassName("out");
     for(var i = 0; i < elements3.length; i++) {
       var v1 = elements3[i].value;
       var v2 = elements4[i].value;
      var text="case = Test " + (i+1)+ "\r\n" + "grade reduction =
100%" + "\r\n"
       text2= text2 + text + "input = " + v1 + "\r\n" + "output = " +
v2 + "\r\n";
  }
     download(filename2, text2);
  }, false);
```

Оценяване

Сравняваме файловете с вида на входните и вида на изходните данни за една и съща задача на студент и преподавател и оценяваме проектирането на входа и изхода.

Текстовият файл с брой и тип на данните се сравнява с предварително създаден от преподавателя и се оценява като процент, който се пресмята на база на частното от брой съвпадения и общ брой тестови случаи, зададени от преподавателя.

Оценяването на файла с вида на входните и вида на изходните данни на студента се осъществява като автоматично се сравнява с файл, подготвен от преподавателя. Двата файла при избор от твърд диск или флаш памет трябва да са подредени в следния ред – първо файла на студента и след това файла

на преподавателя. Двата файла се избират едновременно чрез клавиши Ctrl или Shift.

Процедура за прилагане

1. Студентът записва двата файла създадени и изтеглени чрез функцията „проектиране“ в своя профил на системата за електронно обучение.

2. Преподавателят подготвя текстов файл по същия начин като студента или използва вече готов.

3. Преподавателят изтегля двата файла, записани от студента в системата за електронно обучение и ги оценява първият чрез функцията „оценяване“, а вторият чрез VPL модула на Moodle(за това тестване е необходимо студентът да е записал и програмен код) и нанася резултатите от оценяването в профила на студента.

Алгоритъм за оценяване на вида на входните и вида на изходните данни

Сортираме двата масива, получени от файла на студента и файла на преподавателя като всеки елемент от масива представлява съответния ред от файла.

Сравняваме всеки от елементите на първия масив с всеки от елементите на втория. При съвпадение изтриваме елемента от втория и прекъсваме съответната итерация като преброяваме съвпадението.

Резултатът от оценяването се получава като точки по следната формула като се закръглява до цяло число: $\text{От} = \frac{\text{брой верни}}{\text{общ брой случаи}} * 100$

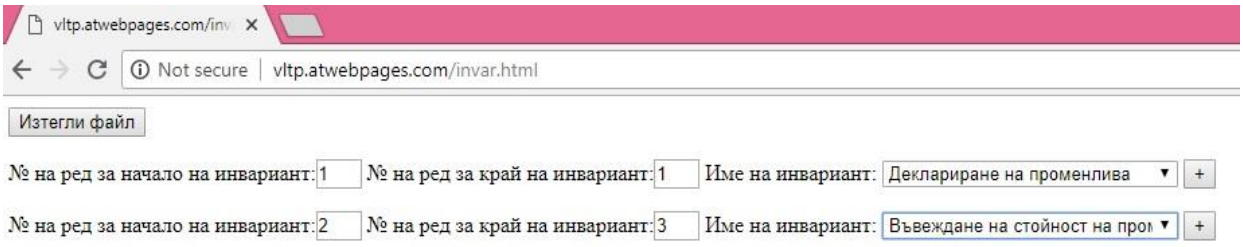
Резултатът от оценяването се получава като оценка по следната формула:

$$2 + \frac{\text{брой верни}}{\text{общ брой случаи}} * 4$$

Функцията, която реализира алгоритъма за **оценяване**:

```
function evaluate(l1,n1,l2,n2){
   var st=l1.valueOf() ;
   st.sort();
   var t=l2.valueOf() ;
   t.sort();
   var count=0;
   for(var ii = 0; ii < n1; ii++)
        for(var j = 0; j < n2; j++)
             if(st[ii]==t[j]){count++;delete t[j];break;}
          document.write(Math.round(count/n2*100),"<br>");
  }
```

Четене с разбиране на програмен код



Фигура 3.2. Екран от сайта за функцията“Четене с разбиране“

Четене с разбиране

Задава се готов програмен код с номерирани редове и потребителят трябва да запише номер на ред за начало на инвариант и номер на ред за край на инвариант и да избере инвариант от предварително зададен списък с наименования на инварианти.

Диапазонът от редове се задава с номерата на началния и крайния ред, а инварианта чрез номенклатурен номер, съответстващ на избраното име от списъка.

В първия текстов файл, който се създава с натискане на бутон, информацията на всеки ред е структурирана по следния начин:

№ на ред за начало на инвариант - **№ на ред за край на инвариант Номенклатурен номер на инвариант**

Вторият текстов файл, който може да се генерира съдържа като текст цялото наименование на инварианта и затова информацията на всеки ред е структурирана по следния начин:

№ на ред за начало на инвариант - **№ на ред за край на инвариант Име на инвариант**

Името на инварианта се избира от следния списък:

- Деклариране на променлива
- Деклариране на масив
- Въвеждане на стойност на променлива
- Въвеждане на стойностите на елементите на масив
- Търсене на минимален елемент
- Търсене на максимален елемент
- Размяна на две променливи
- Сортиране на масив
- Извеждане на стойност на променлива
- Извеждане на стойностите на елементите на масив

Разширено четене с разбиране

Отново се задава готов програмен код с номерирани редове, но допълнително се задават и празни полета, в които да се попълнят стойностите за слотовете на инвариантите. Инвариантът отново се избира от предварително зададен списък и след това за избрания инвариант се попълват стойностите на полетата: **име на променлива, тип на променлива, описание и програмен код**, който го реализира.

Оценяване

Оценяване на файла с номерата на редовете за начало и край на инварианта и кодовете на инвариантите от предварително зададен списък на студента се реализира като автоматично се сравнява с файл, подготвен от преподавателя(създаден по същия начин).

Задачи с инварианти

В Табл. 3.2. е представена БД за описание на инварианти, които се използват в четене с разбиране и задачи с инварианти. БД използва MySQL.

	Номенкл. №	Име на фрейм - инвариант	Име на променлива	Тип на променлива	Описание	Програмен код
Име на поле	Number	InvName	VarName	TypeVar	Description	Code
Тип на поле	SMALLINT	CHAR	TINYTEXT	ENUM	CHAR	LONGTEXT

Таблица 3.2. БД за описание на инварианти

Запознаване с инварианти

В уеб сайта е създадена база от данни за най-често използваните инварианти. При избор на инвариант от предварително зададен списък се извлича пълната информация за него, включително и програмен код.

Модифициране на програмен код

При зададен готов програмен код и условие на нова задача, имаща общ код с първата се предоставя възможност за запълване на форма с полета за задаване на стойностите за слотовете на инвариантите .

Прилага се като се избира инвариант от предварително зададен списък и се попълват стойностите на полетата: име на променлива, тип на променлива, описание и програмен код за новата задача. При избор на инварианта „Деклариране на масив“ в полето описание се задава и максималният брой елементи на масива.

Словесно описание чрез инварианти

Избира се инвариант от предварително зададен списък и се попълват стойностите на полетата: **име на променлива, тип на променлива, описание и програмен код** за задача, зададена с условие и се генерира текстов файл със словесното ѝ описание от зададените стойности. При повече от една променлива в един инвариант – имената на променливите се задават разделени със запетая, а в полето тип на променлива за всяка поотделно се задава тип – както при списък от параметри на функция – напр. double a, double b.

Визуализации на алгоритми

Включва визуализация на определени алгоритми за сортиране чрез JavaScript анимация. Използван е готов програмен код [Визуализация алгоритми], който е модифициран като анимацията за всеки от представените алгоритми е визуализирана отделно. Представени са алгоритмите за сортиране: метод на мехурчето, метод на пряка селекция и бързо сортиране.

3.2. Методика по използване на ВЛОП

За подготвяне на тестовите данни се използва **VLTP**, при което те се съхраняват в текстов файл и могат многократно да се използват след това.

При зададен програмен код проверява дали решава задача с дадено условие като проектира тестови входни и изходни данни, задава ги във формат за VPL модул Moodle и я тества чрез него. За да се реализира този сценарий се използва **VLTP**.

3.2.1. За лекции

Подходящо е и използването на функцията **“запознаване с инварианти“**, която да се приложи при изучаване на нов елемент от езика, свързан с някой инвариант. В този случай е подходящо и използването и на функциите „**модифициране на програмен код**“ за да се проследи как се променя кода, когато се използва новия елемент при решаване на същата задача или при решаване на подобна. Функцията „**словесно описание чрез инварианти**“ е приложима при описание на нов алгоритъм, който включва в себе си няколко от вече изучените инварианти.

3.2.2. За упражнения

VLTP се използва като се прилагат функциите за четене с разбиране и функциите “**запознаване с инварианти**“, “**модифициране на програмен код**“ и “**словесно описание чрез инварианти**“, свързани с решаване на групи от задачи от АФМ. За проектиране на входните и изходни данни също се прилага функцията, реализирана в уеб сайта.

3.2.3. За подготовка за лекции

За подготовка за лекция на тема „Алгоритми за сортиране“, могат да се използват визуализациите на три от тях – метод на мехурчето, метод на пряк избор и бързо сортиране, добавени в **VLTP** като многократно се изпълняват с различна скорост и за различен брой елементи.

3.2.4. За самоподготовка

При решаване на курсова задача за преминаването през различните етапи е необходимо да се използва **VLTP.**

3.2.5. За проверка и контрол

Тестването и оценяването при употребата на VPL модула се извършва чрез използване на скриптове или чрез поредица от контролни примери, записани в зададен формат в текстовия файл "**vpl_evaluate.cases**". Създаването на файлове от този вид може да се реализира чрез използване на функцията „проектиране на вход/изход“ от уеб сайта за **VLTP**.

VLTP се използва и чрез различните функции за оценяване, свързани с оценяване на математическите модели, проектирането на входните и изходни данни и четенето с разбиране.

3.3. Видове задачи за обучение по програмиране

Решаването на задача за програмиране включва реализация на част или на всички десет етапа.

В обучението по програмиране обикновено се решават задачи от следния вид: дадено е условие на задача да се опише алгоритъм по един от трите начина: словесно, чрез блок-схеми или чрез език за програмиране. С въвеждането на АФМ в обучението по програмиране може да се опише словесно и чрез списък от фрейми-инварианти.

Този вид задачи може да се разшири като се премине през всичките десет етапа, описани във втора глава. Изпълняването на тези етапи ще помогне за придобиването на повече знания и умения, свързани с цялостния процес на създаване на програмен продукт, което способства за по-добра практическа реализация впоследствие след приключване на обучението на студента.

За всяка от групите задачи, въведена в обучението по програмиране с въвеждането на АФМ са създадени функции за нейната реализация, а за функцията, свързана с откриване на инварианти и функция за оценяване.

3.4. Изводи

Всяка от групите задачи, въведена в обучението по програмиране с въвеждането на **АФМ** може да се реализира като се използва съответната функция от ВЛОП или някои от инструментите, достъпни и чрез системата за електронно обучение на Тракийски университет.

Установено е, че чрез проектираната VLTP могат да се реализират десетте етапа. Това е резултат и от факта, че в нея е интегриран VPL модула на Moodle, чрез URL има достъп до Snap, Python Tutor и уеб сайт http://vltp.atwebpages.com/, специално създаден за да реализира функции за някои от тези етапи.

Глава 4. Виртуално обучение по програмиране

4.1. За обучението по програмиране

Под **метод на обучение по информатика** във висше училище ще разбираме начина на регулирана взаимосвързана дейност между преподавателя и студента, насочена към достигане на поставените цели на обучението по информатика [Жужжалов, 2004].

Според начина на предаване на информацията от преподавателя към студента методите на обучение са: словесни, нагледни (визуални) и практически [Жужжалов,2004]

Според класификацията на [Оганесян, 1980], методите за обучение са: обяснително-илюстративен (ОИМд), метод на програмираното обучение (МдПгО), евристичен (ЕМдО), проблемен (МдПрО) и метод на моделното обучение (МдМО).

ОИМд се свежда до запомняне на преподавания материал и неговото възпроизвеждане.

МдПгО се изразява в това, че действията на обучаемия са ясни за това, което трябва да направи и тяхната последователност, т.е. те са програмирани или планирани предварително. Изпълнявайки първата част от тези планирани действия, обучаемият преминава към втората част, после след тяхното завършване към третата част и така до получаване на планираните резултати.

Ако при този метод на програмираното обучение са видни междинните резултати, но не е явен метода за постигането им, тогава тази схема на обучението се нарича вече *ЕМдО* като този начин на достигането на планираните резултати се повтаря на всяко междинно ниво чрез евристично търсене.

Когато са известни междинните резултати и пътят до тяхното достигане, тогава може да се стигне до противоречие между тези, които имат значение за решаването на ситуацията и необходимите за случая. Тогава се казва, че това е проблемна ситуация, т.е. търсенето вече има по-сложен характер и се нарича *МдПрО*.

Във всички тези случаи обучаемият знае какви са крайните резултати. До тях се достига с допълнителни самостоятелни извънаудиторни задания, специални форми за търсене на решението и др. В зависимост от разбирането на поставената задача обучаемият получава резултатите и ги сравнява с планираните. Ако не е постигнал крайните резултати отново преминава този път, поради което този метод се нарича *МдМО*. При определени условия може да има съчетания на моделния с евристичния метод.

В табл. 4.1. за всяка форма, подходяща за обучение по програмиране, са представени кои методи от класификацията на Оганесян са приложими.

Форма	Метод на обучение (по Оганесян)				
	ОИМд	МдПгО	ЕМдО	МдПрО	МдМО
Лекции	+	+	-	-	-
Практически упражнения	+	+	+	-	-
Курсова задача	-	-	+	+	+
Самостоятелна извънаудиторна работа	-	+	+	+	+
Виртуална лаборатория	+	+	+	+	+

Таблица 4.1. Прилагани методи на обучение за реализация на съответните форми

Нетрадиционна форма на обучение по програмиране, свързана и с теоретичните познания и с практическите умения е ВЛ. При използване на ВЛОП могат да се реализират и петте метода [Shivacheva & Nedeva, 2016].

4.2. Експеримент 1

Фреймът се използва за представяне на минималната структурирана информация, която дефинира еднотипни явления, събития, ситуации или процеси [Корж, 2016]. В случая на обучение за придобиване на умения и стил по програмиране, фреймови модели са предложени в [Сомова и др., 2014] под името „инварианти“ (пасажи от програмен код, които с малки неструктурни промени се срещат в решенията на задачи за програмиране).

През учебната 2016/2017 година във ФТТ със студенти от първи курс в специалност „Автоматика и компютърни системи“, ОКС „Бакалавър“ (редовно обучение) е експериментирана методика за решаване на задачи от **група Б - Анализ на непознат програмен код, откриване на инварианти (вкл. и на нови), както и на стойностите на техните параметри (съдържание на съответните слотове)**.

Условно, **използваната методика** може да се определи като **„четене с разбиране на програмен код“**, което традиционно се използва в обучението по роден и чужд език. В обучението по програмиране този подход може да се интерпретира като *„разбиране на готов програмен код чрез откриване на фреймове-екземпляри на съответни инварианти за програмиране“*.

За провеждане на експеримента (в дисциплина „Програмиране и използване на компютри“ втора част на ТрЕУ) са подготвени следните учебни материали: таблица от фрейм-инварианти с две колони (наименование на фрейма-инвариант и цвят за маркиране на пасажи с инварианта в програмен текст); таблица за масиви с три колони (тип, име и максимален брой елементи на масива); таблица за променливи и програмен код на C++ (решение на задача по програмиране, условието на която не е известно на студентите).

Задачата на студентите е в програмния код на C++ да открият и маркират (със съответен цвят) фрейми-инварианти, като определят съдържанието на техните слотове. В Табл.4.2. са представени някои от тези фрейми-инварианти [Шивачева и др., 2017].

Наименование на фрейм	Програмен код
Търсене на минимален елемент	m=0; for(i=1;i<n;i++) if(a[i]<a[m])m=i;
Размяна на стойности на две променливи	b=a[m]; a[m]=a[n-1];a[n-1]=b;
Извеждане на елементите на масив	for(i=0;i<n;i++) cout<<a[i]<<endl;

Таблица 4.2. Фрейми-инварианти, открити в програмен код

4.3. Експеримент 2

Втората експериментирана методика, базирана на АФМ, е предназначена за **обучение по решаване на задачи от група Д. - Самостоятелно проектиране и писане на програмен код** (по дадени спецификации) и се свежда до:

- проектиране на решението под формата на редица от фрейми-инварианти (задача от **група Г.**);
- модифициране на съдържанието на слотовете на фреймите-инварианти (спец. на слот „Програмен код“ – умение, което се предполага че е придобито от решаване на задачи от **група В.**) ;
- реализация на програмен код („сглобяване“ на модифицираните пасажи от програмен код).
- тестване и коригиране.

4.4. Експеримент 3

Във ФТТ по време на практически упражнения по дисциплината „Алгоритми и структури от данни“ със студенти от втори курс редовно обучение на специалност „Автоматика и компютърни системи“ през учебната 2017/2018 год. са експериментирани решаване на задачи от **група В** - **Модифициране на непознат програмен код** (с промяна на съдържанието на слотовете на фрейми-инварианти и въвеждане на нови инварианти)и на база на тях решаване на задачи от **група Д** - **Самостоятелно проектиране и писане на програмен код (по дадени спецификации)**.

Условията на двете задачи за програмиране са:

Условие на програмния код, който е зададен: Програма, която разменя минимален и първи елемент в едномерен масив с n елемента.

Условие на програмата за модифициране: Програма, която разменя минимален и максимален елемент в едномерен масив с n елемента [Шивачева и Тотков, 2017].

4.5. Експеримент 4

Във ФТТ по време на практически упражнения по дисциплината „Програмиране и използване на компютри - втора част“ със студенти от първи курс редовно обучение на специалност „Автоматика и компютърни системи“ през учебната 2017/2018 год. при запознаване с условен оператор в езика за програмиране C++ са използвани визуалните средства **FlowProgramming, Flowgorithm, Snap и Python Tutor** за реализиране на алгоритми за намиране на по-голямото от две числа, за решаване на линейно уравнение от вида:**ax+b=0** и квадратно уравнение от вида:$\mathbf{ax^2+bx+c=0}$ [Шивачева, 2018].

4.6. Изводи

Методите за обучение от разгледаните класификации са подходящи и могат да се прилагат в обучението по програмиране като се използват ВЛОП и представените средства за визуализация.

Заключение

В рамките на дисертационното изследване са решени следните основни задачи:

Задача 1. Да се направи анализ на състоянието на изследванията в областта на виртуалните лаборатории и по-конкретно на виртуалните лаборатории за обучение по програмиране (ВЛОП).

Задача 2. Да се определят функционалните изисквания към среда от тип „ВЛОП“ и да се проектират концептуален и функционален модел, архитектура и бази от данни.

Задача 3. Да се реализира прототип на среда за ВЛОП.

Задача 4. Да се организират и проведат експерименти за виртуално обучение по програмиране.

С решаването на задачи 1. – 4. е **постигната основната цел** на дисертационното изследване – проектиране и реализиране на виртуална лаборатория за обучението по програмиране, която да даде възможност за повишаване на качеството на учебния процес при традиционното обучение, да подпомогне самостоятелната подготовка на студентите и да осигури условия за дистанционно обучение.

Приноси на дисертационния труд

Основните приноси на дисертационния труд могат да се характеризират като научни, научно-приложни и приложни.

Научни приноси на дисертационното изследване са:

Н1. Създадени са фреймови модели (фрейми-прототипи), подходящи за автоматизиране на обучението по програмиране;

Н2. Предложен е концептуален модел на среда за виртуално обучение по програмиране;.

Н3. Създадена е методика за обучение по програмиране от тип „четене с разбиране“.

Научно-приложни приноси на дисертационното изследване са:

НП1. Определени са функционалните изисквания към среда за ВЛОП;

НП2. Проектиран е концептуален и функционален модел на среда за ВЛОП.

Приложни приноси на дисертационното изследване са:

П1. Създаден е софтуерен прототип на среда от тип ВЛОП;

П2. Софтуерният прототип на ВЛОП е експериментиран на базата на решаване от обучавани на задачи, типични за началното обучение по програмиране;

П3. Подготвени, проведени и анализирани са експерименти в обучението по програмиране, свързани с прилагане на методиката за „четене с разбиране на програмен код“ ;

П4. Реализиран е експеримент по автоматично тестване на студентски програми във VPL модула на Moodle.

Перспективи за развитие

Могат да се посочат следните перспективи за развитие:

- Автоматично **генериране на програмен код** на база на словесно описание с инварианти;
- Проектиране на **системи от типови АФМ** (за различни теми, дидактически единици и др.);
- Изследване на алгоритми и методи за **извличане и агрегиране на данни от учебни материали** в процеси на е-обучение, „маркирани“ с АФМ;
- Разработването на **методики за използване на АФМ** в е-обучението.

Използвана литература

[Жужжалов, 2004] Жужжалов В.Е. Специфика обучения программированию при подготовке студентов-информатиков //Вестник МГПУ. Сер. «Информатика и информатизация образования».– М., 2004, № 1 (2), с. 56–61

[Корж, 2016] Корж, Т.Н., Обучение переводу русскоязычных художественных текстов с опорой на жанровый фрейм, DOI: 10.17223/19996195/36/5, 2016

[Оганесян, 1980] Оганесян В.А. и др. Методика преподавания математики в средней школе: Общая методика. Учебное пособие для студентов физ.-мат. фак. пед. ин-тов /В. А. Оганесян, Ю. М. Калягин, Г. Л. Луканкин, В. Я. Саннинский. -2-е изд., перераб. и доп. —М.: Просвещение, 1980. -368 с.: илл.

[Постановление №122 на МС, 2018] Постановление № 122 от 29 юни 2018 г. за изменение и допълнение на *Правилника за прилагане на Закона за развитието на академичния състав в Република България*, приет с Постановление № 202 на Министерския съвет от 2010 г., Държавен вестник, бр.56, 06.07.2018г., стр.18-34

[Смрикаров, 2009] Смрикаров, А., Националната програма за създаване на Виртуално Образователно Пространство основни резултати и предстоящи задачи младежкото участие и принос ISBN 978-954-712-403-5, 2009г.

[Сомова и др., 2014] Сомова Е., Й. Енев, Г. Тотков, Инварианти в обучението по програмиране, International scientific on-line journal Natural & Mathematical science, Vol. 4, No. 3 (2014).

[Томов, 2011] Томов, О. ,дисертация "Създаване и изследване на виртуални лаборатории по компютърни системи и технологии", 2011

[Тотков и др., 2014в] Тотков Г. и др., Методика на е-обучението (ред. Г. Тотков), „Ракурси", Пловдив, 2014, 978-954-8852-43-2.

[Тотков и др., 2014г] Тотков Г. и др., Съвременни тенденции в е-обучението (ред. Г. Тотков), „Ракурси", Пловдив, 2014, ISBN 978-954-8852-46-3.

[Трухин, 2003] Трухин , А. В. , Виртуальные компьютерные лаборатории: классификация, Томский государственный университет систем управления и радиоэлектроники, 2003г., http://ido.tsu.ru/files/pub2003/2003_Truhin.pdf

[Шивачева и др., 2015] Шивачева, Г., В. Недева, М. Василева, Д. Георгиева, Софтуер за изграждане на виртуални лаборатории, XXIV Международна научна конференция „Мениджмънт и качество" за млади учени, 15 – 16 октомври 2015 г., Ямбол, Сборник научни трудове стр.292-300

[Шивачева и др., 2017] Шивачева Г., Г. Тотков, Р. Донева. Четене с разбиране на програмен код, Научни трудове на СУБ – Пловдив, Серия В. Техника и технологии, том. XV, стр. 58-62, ISSN 1311-9419 (Print) ISSN 2534-9384 (On-line)

[Шивачева и Тотков, 2017] Шивачева Г., Г. Тотков, Акумулативни фреймови модели в обучението по програмиране, Proceedings of International Conference on Technics, Technologies and Education ICTTE 2017 October 19-20 2017, Yambol, Bulgaria, ISSN 1314-9474, pp.277-283

[Шивачева, 2018] Шивачева Г., Изучаване на разклонени алгоритми чрез средства за визуализация, Сборник доклади от Международна конференция за млади учени"Мениджмънт и качество" – Ямбол 10-12.05.2018 (в печат)

[Bose, 2012] Bose, Dr. Ranjan, Professor, IIT Delhi led the Weekly Discussion via A-VIEW with this talk on "Virtual Labs" on 31st October 2012.

[Harms, 2000] Harms , U., Virtual And Remote Labs In Physics Education, German Institute for Research on Distance Education at the University of Tuebingen, 2000

[Prieto-Blazquez, 2009] Prieto-Blázquez, J., J. Herrera-Joancomartí, A Virtual Laboratory Structure for Developing Programming Labs, doi:10.3991/ijet.v4s1.789, iJET, 2009, Vol.4, pp.47-52, ISSN:1863-0383

[Shivacheva & Nedeva, 2016] Shivacheva, G., Nedeva, V., Methods for Teaching Programming Using Virtual Laboratory, Proceedings of the 11th International Conference on Virtual Learning, Bucharest, 29.10.2016, ISSN 1844-8933, pp. 92-98

[Shivacheva et al, 2017] Shivacheva G., V. Nedeva, Sv. Atanasov, Designing a Virtual Laboratory for Teaching Programming, 18-th International Conference on Computer Systems and Technologies CompSysTech'17 , June 2017, Proceedings, ACM ICPS Vol.1369, ISBN 978-1-4503-5234-5, pp 310-317

[Zhou et al, 2005] ZHOU YI, JIANG JIAN-JUN and FAN SHAO-CHUN, A LabVIEW-Based, Interactive Virtual Laboratory for Electronic Engineering Education, 2005

[Визуализация алгоритми] http://jsdo.it/norahiko/oxly, последен достъп на 05.07.2017

[Flowgorithm] http://flowgorithm.org/, последен достъп на 02.05.2018г.

[FlowProgramming] https://www.technoprogram.com/flowchart-visual-programming-3-01-turkce-indir.html, последен достъп на 02.05.2018г.

[PythonTutor] http://www.pythontutor.com/, последен достъп на 02.05.2018г.

[Snap] http://snap.berkeley.edu/, последен достъп на 02.05.2018г.

[VPL] http://vpl.dis.ulpgc.es/, последен достъп на 06.10.2016г.

Списък с публикации

по темата на дисертационния труд

1. **Г. Шивачева**, В. Недева, М. Василева, Д. Георгиева, Софтуер за изграждане на виртуални лаборатории, XXIV Международна научна конференция „Мениджмънт и качество" за млади учени, 15 – 16 октомври 2015 г., Ямбол
2. **Shivacheva, G.**, Nedeva, V., Bochev, V., Dimova, E., An Innovative Approach For Training In Programming Using Virtual Laboratories, Международна научна конференция "Техника, технологии, образование" ICTTE 2016, Ямбол, 17-18 ноември 2016, ARTTE Vol. 4, No. 3, 2016, pp. 203-210, ISSN 1314-8788 (print), ISSN 1314-8796 (online)
3. **Шивачева, Г.**, Преимущества применения виртуальных лабораторий в обучении программированию, Материалы V международной научной конференции молодых ученых, аспирантов, студентов, магистрантов „Актуальные проблемы моделирования, проектирования и прогно- зирования социальных и политических процессов в мультикультуральном пространстве современного общества", Ростов на Дон, 4-8 април 2016, ISBN 978-5-9908135-7-1, стр.168-179
4. **Шивачева, Г.**, Недева, В., Съвременни аспекти на обучението по програмиране с виртуална лаборатория, Девета национална конференция"Образованието и изследванията в информационното общество", Сборник доклади на Национална конференция, Пловдив, 26-27 май 2016, ISSN 1314-0752, стр.197-206
5. **Shivacheva, G.**, Nedeva, V., Methods for Teaching Programming Using Virtual Laboratory, Proceedings of the 11th International Conference on Virtual Learning, Bucharest, 29.10.2016, ISSN 1844-8933, p. 92-98
6. Nedeva, V., **Shivacheva, G.**, Educational Technology Using Virtual Laboratory for Teaching Programming, Proceedings of the 11th International Conference on Virtual Learning, Bucharest, 29.10.2016, ISSN 1844-8933, p. 192-198
7. **Шивачева Г.**, Г. Тотков, Р. Донева. Четене с разбиране на програмен код, Научни трудове на Съюза на учените в България – Пловдив, Серия В. Техника и технологии, том. XV, 2017, с.58-62

8. **Шивачева Г.**, Г. Тотков, Акумулативни фреймови модели в обучението по програмиране, Proceedings of International Conference on Technics, Technologies and Education ICTTE 2017 October 19-20 2017, Yambol, Bulgaria, ISSN 1314-9474, pp.277-283
9. **Shivacheva G.**, V. Nedeva, Sv. Atanasov, Designing a Virtual Laboratory for Teaching Programming, 18-th International Conference on Computer Systems and Technologies CompSysTech'17 , 23-24 June 2017, University of Ruse, Bulgaria, ACM ICPS Volume 1369, with ACM ISBN 978-1-4503-5234-5, pp. 310-317, реферирана в Scopus
10. **Шивачева Г.**, Изучаване на разклонени алгоритми чрез средства за визуализация, Сборник доклади от Международна конференция за млади учени"Мениджмънт и качество" – Ямбол 10-12.05.2018(приет).

Забелязани цитирания

на публикации по темата на дисертационния труд

1. Г. Шивачева, В. Недева, М. Василева, Д. Георгиева, Софтуер за изграждане на виртуални лаборатории, XXIV Международна научна конференция „Мениджмънт и качество" за млади учени, 15 – 16 октомври 2015 г., Ямбол, Сборник научни трудове стр.292-300

Цитиращи автори и публикации/доклади:

1) Zlatev, Zl., Design Of Laboratory Measurement Equipment For Temperature And Humidity Considering The Possibility Of Application In Distance Learning, IInnovation and entrepreneurship, ISSN 1314-9253 Volume IV, number 3, 2016, pp.16-27
2) Zlatev, Z., A. Dimitrova, S. Baycheva, M. Vasilev, Analysis of information processes in the production of yogurt, Innovation and entrepreneurship, Applied scientific journal, Vol.4, No.2, ISSN 1314- 253, 2016, pp.43-59
3) Vasilev, M., Classification Of Yellow Cheese In Storage Period By Nonlinear Discriminant Analysis And Color Features, Innovation and entrepreneurship, ISSN 1314-9253, Volume IV,number 3, 2016, pp.28-37
4) Dimitrova I., I. Dimov, M. Zhelyazkova, E-learning Course "Physical and Colloidal Chemistry" at the universities, Proceedings of the 11th International Conference on Virtual Learning, Bucharest, 29.10.2016, ISSN 1844-8933, pp.99-103
5) Baycheva S., Z. Zlatev, A. Dimitrova, Investigating the possibilities of document cameras for quality assessment of foodstuffs by measuring of color, Proceedings of the 11th International Conference on Virtual Learning, Bucharest, 29.10.2016, ISSN 1844-8933, pp.204-208
6) Elnashar, E. A., Z. Zlatev, P. Boneva, Education in the professional field "Design": A comparative analysis of Egyptian and European experience in this study area of higher education, International Journal of Advanced Educational Research,

ISSN: 2455-6157; Impact Factor: RJIF 5.12, Vol. 2; Issue 2; March 2017; pp 5-9

2. Shivacheva G., V. Nedeva, Modern aspects of training in programming with virtual laboratory, Education and research in the information society. Proceedings of the National Conference (Plovdiv 26-27 May 2016). ISSN 1314-0752, pp. 197-206 (in Bulgarian).

Цитиращи автори и публикации/доклади:

1) Stoykova, V., Zl. Zlatev, St. Baycheva, Possibilities for application of document cameras and webcams in the lecture halls and laboratories of the universities, ARTTE Vol 4 No 2, 2016, pp. 125-132, ISSN 1314-8788 (print), ISSN 1314-8796 (online)
2) Zlatev, Zl., M. Baeva, P. Nikolova, Segmentation Of Microscopic Images Of Bacteria In Bulgarian Yoghurt By Template Matching, ARTTE Vol. 4, No. 3, 2016, pp. 211-225, ISSN 1314-8788 (print), ISSN 1314-8796 (online)