МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
федеральное государственное бюджетное образовательное учреждение высшего образования
КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ
им.В.П.АСТАФЬЕВА
(КГПУ им.В.П.Астафьева)

Институт/факультет Математики, физики и информатики
(полное наименование института/факультета/филиала)

Выпускающая(ие) кафедра(ы) Базовая кафедра Информатики и информационных технологий в образовании
(полное наименование кафедры)

Ковач Александра Игоревна

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема: Двумерная графическая среда обучения программированию на языке Си/Си++ в старшей школе

Направление подготовки 44.03.05 Педагогическое образование
(код направления подготовки)

Профиль Информатика
(наименование профиля для бакалавриата)

ДОПУСКАЮ К ЗАЩИТЕ
Зав.кафедрой д.п.н., профессор Пак Н.И.
(ученая степень, ученое звание, фамилия, инициалы)
11.06.2018 Романов
(дата, подпись)

Руководитель к.ф.-м.н., доцент базовой кафедры ИИТО, Романов Д.В.
(ученая степень, ученое звание, фамилия, инициалы)

Дата защиты 22.06.2018
Обучающийся Ковач А.И.
(фамилия, инициалы)
11.06.2018
(дата, подпись)

Оценка хорошо
(прописью)

Красноярск
2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
федеральное государственное бюджетное образовательное учреждение высшего образования
КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ
им.В.П.АСТАФЬЕВА
(КГПУ им.В.П.Астафьева)

Институт/факультет Математики, физики и информатики
(полное наименование института/факультета/филиала)

Выпускающая(ие) кафедра(ы) Базовая кафедра Информатики и информационных технологий в образовании
(полное наименование кафедры)

Ковач Александра Игоревна

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема: Двумерная графическая среда обучения программированию на языке Си/Си++ в старшей школе

Направление подготовки 44.03.05 Педагогическое образование
(код направления подготовки)

Профиль Информатика
(наименование профиля для бакалавриата)

ДОПУСКАЮ К ЗАЩИТЕ
Зав.кафедрой д.п.н., профессор Пак Н.И.
(ученая степень, ученое звание, фамилия, инициалы)

(дата, подпись)
Руководитель к.ф.-м.н., доцент базовой кафедры ИИТО, Романов Д.В.
(ученая степень, ученое звание, фамилия, инициалы)

Дата защиты _____
Обучающийся _____
(фамилия, инициалы)

(дата, подпись)
Оценка_____
(прописью)

Красноярск
2018

**ОГЛАВЛЕНИЕ**

## ВВЕДЕНИЕ

Мы живем в эпоху, когда все развивается и меняется с невероятной скоростью. В эпоху инноваций, как порой называют наш век. И система российского образования не остается в стороне, стараясь «шагать в ногу со временем». Перемены, развитие, новые технологии – все это влечет за собой пересмотр и корректировку устоявшихся ранее школьных программ, методик обучения, результатов обучения в общеобразовательных школах. Но, несмотря на это, в некоторых направлениях система лишь пытается догнать завтрашний день.

*«…некоторое минимальное множество понятий и методов, достаточное для формирования представления о современной информационной реальности и современной информатике, …курс школьной информатики должен быть частью информатики «настоящей».* [15]

Темп жизни и доступность огромного количества обучающих ресурсов и источников информации привели к тому, что обострился конфликт между учебными задачами и требованиями индустрии разработки. В настоящее время в 10-11 классах, когда изучается тема алгоритмизации и программирования, обучение происходит на языке Pascal. Это один из наиболее известных в школе языков программирования. Он легок в изучении, создано множество апробированных методик и литературы, сам язык ясен, компактен и выразителен. Тем не менее, как язык разработки Pascal был актуален в основном в 90-ые годы благодаря изумительно быстрому компилятору фирмы Борланд, просто не имеющему конкурентов по скорости разработки, и входящему всего на одну дискету 3.5 дюйма. В настоящее время он занимает 0.875% (меньше процента) рынка разработки и почти не востребован, используясь в небольшом сегменте индустрии (Си-подобные языки занимают больше 48% всего рынка разработки: одни только Си и Си++ занимают первое и второе места рейтинга, 14.9% и 8.34%

соответственно). Многие новые технологии развиваются, одновременно формируя экосистему инструментов и пользователей: машинное обучение и искусственный интеллект - Питон и Си++; робототехника - Питон, Си, Си++. Pascal используется только в качестве нового языка Оберон в узком сообществе физиков высоких энергий, в отдельных закрытых областях (банковский сектор).

Сейчас, когда важна поддержка, скорость написания и развёртывания, помощь консультантов и коллег, программы, написанные на данном языке, оказываются тяжким грузом для исполнителя из-за малости доли рынка и отсутствия доступного сообщества, которая занимается разработкой на Pascal профессионально. Тем не менее, большинство источников (ссылка на Quora выше и остальные) отмечают замечательно пологую кривую обучения языку, обусловленную доступной средой разработки, работающей из коробки, простоте синтаксиса и возможности начинать обучение с малых и понятных примеров. Именно поэтому целью моего исследование стало создание среды для обучения программированию на языке Си/Си++ в старшей школе, обладающей сходными достоинствами и, дополнительно, опирающейся на наработки отечественной школы обучения информатике с использованием графических исполнителей.

Таким образом, хотелось бы, что бы упор в школьной программе шел на более востребованные языки, которые зачастую даже поверхностно не рассматриваются. Мною были выбраны языки Си и Си++, как одни из самых востребованных на рынке труда, что решало заодно вопросы мотивации сильных учеников.

В качестве объекта исследования выбран процесс обучения программированию в старшей школе, предметом - когнитивные сложности при первоначальном знакомстве с базовыми конструкциями процедурных компилируемых языков программирования.

В качестве основных задач исследования, стояли:

1. Анализ причин успешности языка Pascal.
2. Анализ основных трудностей при обучении языкам Си и Си++ как первым языкам программирования, особенно самостоятельно.
3. Создание среды разработки, сочетающей синтаксис языков Си и Си++ с удачными решениями, обеспечивающими популярность языку Pascal.
4. Разработка двумерной графической среды с исполнителем для развития алгоритмического мышления и обучению программированию на языке Си/Си++.

# ГЛАВА 1. Методические требования и анализ положения языка программирования в школьном курсе информатики и смежных областях

## 1.1. Анализ школьного курса информатики

Информатика как образовательная дисциплина быстро развивается. Если раньше базовый курс информатики состоял из изучения основ алгоритмизации и программирования, основ устройства и применения вычислительной техники, сети, кодирования и т.д., то сегодня целью курса информатики в школе является повышение эффективности применения человеком компьютера как инструмента решения задач науки, производства и бизнеса. Компьютерная грамотность определяется не только умением программировать, но и умением использовать готовые программные продукты, рассчитанные на пользовательский уровень, сочетая различные решения благодаря развитому алгоритмическому мышлению.

Эта тенденция появилась благодаря широкому рассмотрению продуктов, ориентированных на неподготовленных пользователей. Например, мобильные справочники, голосовые переводчики, обучающие приложения и т.д. Разработка таких программно-информационных средств является весьма дорогостоящим делом в силу его высокой наукоемкости и необходимости совместной работы высококвалифицированных специалистов: психологов, компьютерных дизайнеров, программистов. Однако она окупает себя благодаря тому, что доступ к компьютеру сегодня может получить практически каждый человек даже без специальной подготовки.

Содержание курса информатики включает совокупность двух взаимосвязанных компонентов: теоретического и практического. Теоретическая часть курса направлена на формирование основ информационной культуры, навыков анализа и формализации предметных

задач, ознакомление с такими понятиями как информация, сообщение, свойства информации, информационные процессы, алгоритм, исполнитель алгоритма, структура алгоритма, величина, типы величин. Практический аспект связан с выработкой навыков работы с готовым программным обеспечением, написанием программ на одном из конкретных языков программирования, использованием глобальной сети Интернет для обмена информацией и сообщениями, ее поиска. Необходимость выработки практических навыков и умений работы на компьютере предусматривает значительное повышение удельного веса практических занятий (по сравнению с другими предметами) в общей структуре курса информатики предоставляя специфические черты, которые отличают его от других предметов.

Развивающая цель реализуется в процессе овладения учащимися опытом творчества поисковой деятельности, осознания явлений окружающей действительности, их сходства и различия, что предполагает развитие у учащихся:

- логического мышления и интуиции, пространственного воображения,
- умений переноса знаний и навыков в новую ситуацию на основе осуществления проблемно-поисковой деятельности,
- интеллектуальных и познавательных способностей (разных видов памяти – слуховой и зрительной, оперативной и долговременной внимания - произвольного и непроизвольного, воображения и т.п.),
- готовности к овладению и использования новой компьютерной техники и нового программного обеспечения,
- готовности к дальнейшему самообразованию в области информационных и технологий.

Новые понятия и методы, изучаемые в курсе информатики, существенно расширяют традиционные границы школьного математического инструментария, формируют новые прикладные знания, умения и навыки, подводят любознательного школьника к пониманию фундаментальных методов современной науки и их применения на практике.

Понятийный аппарат информатики включает универсальные понятия, которые достаточно широко используются в других науках и в повседневной практике людей (объект, субъект, модель, информация, сообщения, алгоритм, система, схема кодирования, передачи информации и т.п.) и узкоспециальные, без которых невозможна успешная работа на компьютере (операционная система, файл, драйвер, отладка программы, прерывания и т.д.).

Задачи, которые решаются в рамках курса информатики, часто принадлежат к другим областям предметных знаний – математике, физике, химии, биологии, истории и др.

Следующие особенности курса информатики, выделяют его из других школьных предметов: динамичность содержания курса школьной информатики; отсутствие общепринятого среди учителей понимания информатики как науки и как учебного предмета; неоднозначность понимания целей обучения; разнообразие ориентаций в действующих учебниках; тенденция к интеграции школьного образования; тенденция к снижению возраста обучения информатике и др.

В настоящее время существует несколько подходов к преподаванию основ программирования в школе. Один из них – преподавание языков программирования высокого уровня на базе конкретной системы программирования.

В стандарте по информатике отмечается, что в результате изучения информатики на базовом уровне ученик в области программирования должен:

1. знать основные свойства алгоритмов, типы алгоритмических конструкций: прохождение, разветвления, цикл, понятие вспомогательного алгоритма;
2. уметь использовать алгоритмические конструкции, выполнять и строить простые алгоритмы, выполнять базовые операции над объектами: цепочками символов, числами, списками, деревьями;
3. использовать приобретенные знания и умения в практической деятельности и повседневной жизни при выполнении индивидуальных и коллективных проектов, в учебной деятельности, в дальнейшем освоении профессий.

Данные знания, умения и навыки формируются при изучении темы «Алгоритмизация и программирование».

Программирование – одна из самых традиционных тем в курсе информатики, но место и вес этой темы в программе данной дисциплины со временем изменяются. Наиболее существенный пересмотр этого вопроса состоялся с переносом информатики из старших классов в базовую школу и с развитием компьютерных технологий.

Новым направлением в развитии темы «Программирование» является введение в курс информатики изучения объектно-ориентированных языков программирования. Учитель информатики может в зависимости от наличия компьютерной техники и профильного направления школы преподавать структурный или объектно-ориентированный язык программирования. Как объектно-ориентированный язык, в школьной практике можно использовать

расширение языка Pascal – среда визуального программирования Delphi-Pascal и Visual Basic.

Программирование – это раздел информатики, задача которого – разработка программного обеспечения ЭВМ. В узком смысле слово программирование означает процесс разработки программы на определенном языке программирования. Разработку средств системного программного обеспечения и систем программирования принято называть системным программированием; разработку прикладных программ называют прикладным программированием. Согласно этому принципу разделяют программистов на системных и прикладных, в зависимости от типа создаваемых программ.

Обучение составлению алгоритмов работы с величинами должно проводиться на примерах типовых задач с постепенным усложнением структуры алгоритмов. Последовательность задач планируется согласно сложности и последовательности изучения алгоритмических структур:

- линейные алгоритмы (вычисления по формулам, любые пересылки значений переменных);
- разветвленные алгоритмы (поиск наибольшего или наименьшего значений из нескольких данных, сортировки 2-3 значений, диалог с разветвлениями);
- циклические алгоритмы (вычисления сумм и произведений числовых последовательностей, циклическое ввода данных с последующей их обработкой).

Изучение языка программирования Pascal происходит в контексте решаемых задач, то есть новые средства языка вводятся по мере необходимости для решения очередного типа задач.

Основной метод изучения – демонстрация языка на примерах простых программ с краткими комментариями. Некоторые понятия достаточно воспринимать ученикам на интуитивном уровне. Наглядность такого языка как Pascal облегчает это восприятие. Пониманию материала помогает аналогия между Pascal и английским языком. Для выполнения учениками несложных самостоятельных задач на первом этапе достаточно действовать методом «по образцу».

Проблему как связать изучение методов построения алгоритмов работы с величинами и языка программирования можно решить 2 вариантами:

1. Сначала рассматриваются различные алгоритмы, для описания которых используются блок-схемы и алгоритмический язык, а затем – правила языка программирования, способы перевода уже построенных алгоритмов в программу на этом языке.
2. Алгоритмизация и программирование осваиваются параллельно.

Желательно, чтобы учащиеся как можно раньше получили возможность проверять правильность своих алгоритмов, работая на компьютере. Но даже при использовании компьютера, прежде всего, рекомендуется не отказываться от ручной трассировки.

На сегодняшний день актуальным стало олимпиадное движение. Школьники со всей России принимают участие во всероссийской инженерной олимпиаде – НТИ[36], Wordskills, «Технокубок», «Ломоносов» и в множестве иных региональных, всероссийских и международных олимпиадах по информатике. Они предусматривают решение учениками олимпиадных задач по программированию.

**Задача 1.3.2 (2 балла)**

*Условие:*

В логистическом центре 5 роботов-погрузчиков занимались погрузкой ящиков с грузами. Первый робот перетащил ровно A% от всех ящиков, второй робот — ровно B% от всех ящиков, третий робот — ровно C%, четвертый робот — ровно D%, пятый робот перетащил ровно E%.

Какое минимальное количество ящиков было погружено роботами?

**Формат входных данных:**

Единственная строка входных данных содержит пять чисел A, B, C, D, E, $(A + B + C + D + E = 100, 0 \leq A, B, C, D, E \leq 100)$

**Формат выходных данных:**

Выведите минимальное количество погруженных ящиков.

**Пример:**

***stdin:***
15 55 0 5 25
***stdout:***
20

*Решение:*

Очевидно, что минимальное количество ящиков не превысит 100, это просто A+B+C+D+E. То есть достаточно перебрать возможные ответы и проверить их на правильность, среди них выбрать минимальный.

Пример программы, реализующей данный алгоритм на языке Python:

```
import sys

def solve(dataset):
    a = [int(x) for x in dataset.split()]
    for i in range(1, 101):
        failed = False
        for x in a:
            if i * x % 100 != 0:
                failed = True
        if not failed:
            return str(i)
    return str(100);

print(solve(sys.stdin.read()))
```

Рисунок 1. Задача из Олимпиады НТИ.

Роль олимпиад значительно увеличилась, так как после отмены льгот медалистов при поступлении в вузы, победа в олимпиадах является главным показателем качества знаний учащихся, дает им право на льготное поступление. Поэтому учителям информатики необходимо обучать программированию, чтобы направлять детей к участию в олимпиадах по информатике. А для достижения высокого уровня усвояемости материала, необходимо строит процесс обучения на дидактических принципах:

- Принцип сознательности и активности, который заключается в использовании разнообразных приемов, способствующих возбуждению потребности и интереса к овладению знаниями. Если задачи однотипны и не имеют никакого развития, то учащийся просто потеряет интерес к обучению.
- Принцип наглядности. Наглядность обучения выступает для учащихся средством познания окружающего мира, то есть на непосредственном наблюдении и изучении предметов, явлений или событий. По мнению Ушинского, наглядное обучение повышает внимание учащихся, способствует более глубокому усвоению знаний, ведь мышление детей развивается от конкретного к абстрактному.

Подобный подход к процессу обучения, не только упрощает его, но и повышает интерес учащихся к знаниям и делает процесс обучения более легким. Ведь куда занятнее тот процесс, где ты, совершив какое-либо действие, вносишь изменении. А для детей, сложно найти что-то нагляднее чем игра. Что реализовано на сайте code.org.

### 1.2. Анализ достоинств языка Pascal как первого языка программирования в старшей школе

*«Программисты — это зачастую яркие люди, которые гордятся <…> своей способностью справляться со сложностями и ловко обращаться с абстракциями. Часто они состязаются друг с другом, пытаясь выяснить, кто может создать «самые замысловатые и красивые сложности». <…> соперники полагают, что должны соревноваться с чужими «украшательствами» путём добавления собственных. Довольно скоро «массивная опухоль» становится индустриальным стандартом, и все используют большие, переполненные ошибками программы, которые не способны удовлетворить даже их создателей. <…>*

*<…> такой подход может обернуться неприятностями, если программисты реализуют простые вещи сложными способами, просто потому что им известны эти способы, и они умеют ими пользоваться.»* [20].

Успешность современного процесса обучения в значительной степени зависит от удачного выбора средств обучения. Информационно-коммуникационные средства обучения вместе с живым словом педагога являются важным компонентом учебно-воспитательного процесса и обязательным элементом учебно-материальной базы образовательного учреждения. Совершенные и эффективные средства обучения – залог результативности образования. Использование традиционных средств обучения, подкрепленное большим опытом, использование компьютерных программ как средств обучения все еще бурно развивается. Сейчас определенную специфику имеет ознакомление учащихся общеобразовательных учебных заведений с технологиями создания программного обеспечения. В процессе изучения программирования основным средством обучения выступают исполнитель и его система команд (язык программирования).

В начале компьютерной эры программисты были зависимы от вычислительных машин. На тот момент, единственный язык, который понимал компьютер – двоичный код. С течением времени, программирование требовало все больше времени, а внесение изменений в программы и их модернизация становились практически невозможными. Тогда стали появляться первые языки программирования высокого уровня, то есть отличные от машинного кода, одним из них был Pascal.

Pascal – язык со строгой типизацией и наличием средств структурного программирования, созданный в 1968-1970 г.г. в Швейцарском Федеральном институте технологии в Цюрихе Никласом Виртом. Автор считал, что язык

должен способствовать дисциплинированному программированию. Благодаря этому Николас заложил в свое детище не только строгую типизацию, но и свел к минимуму возможные синтаксические неоднозначности, что является в настоящее время определяющей характеристикой языков программирования. По сравнению с основным языком программирования в академической среде 1970-х, Pascal представлял собой значительный шаг вперед. К 1980-м годам он стал основой для многочисленных учебных программ, получив широкое распространение после появления удобного и эффективного компилятора Turbo Pascal.

Именно с Pascal впервые в программировании появились и принципиально новые, на тот период, типы данных, что сказалось на самом стиле разработки программ.

1. В настоящее время Pascal остается почти идеальным для изучения в качестве первого языка программирования и, по большей части, единственным, что дети изучают по школьной программе в старших классах. На слуху следующие достоинства языка Pascal:
2. многолетние, апробированные методики обучения;
3. язык легок в изучении;
4. используются понятные английские слова (begin, end, or, and, not);
5. строгая типизация - развивает навыки чтения программного кода;
6. большое количество компиляторов под большинство ОС (кроссплатформенность);
7. внятные языковые конструкции: отсутствуют операторы, способные изменять значения переменных внутри выражений;
8. достаточно низкие аппаратные и системные требования, как самого компилятора, так и программ, написанных на Pascal;

9. поддержка структурного программирования, а также объектно-ориентированного программирования.

Рассмотрим процесс обучения внимательнее.

**Шаг 1:** процесс работы с языком Pascal начинается с загрузки и запуска среды разработки. Большинство сред (IDE) изначально поставлялись сразу вместе с компилятором, начиная с Turbo Pascal (1983 год). Современная среда, наиболее хорошо зарекомендовавшая себя в школе: PascalABC. Оба интерфейса показаны ниже на рисунках:

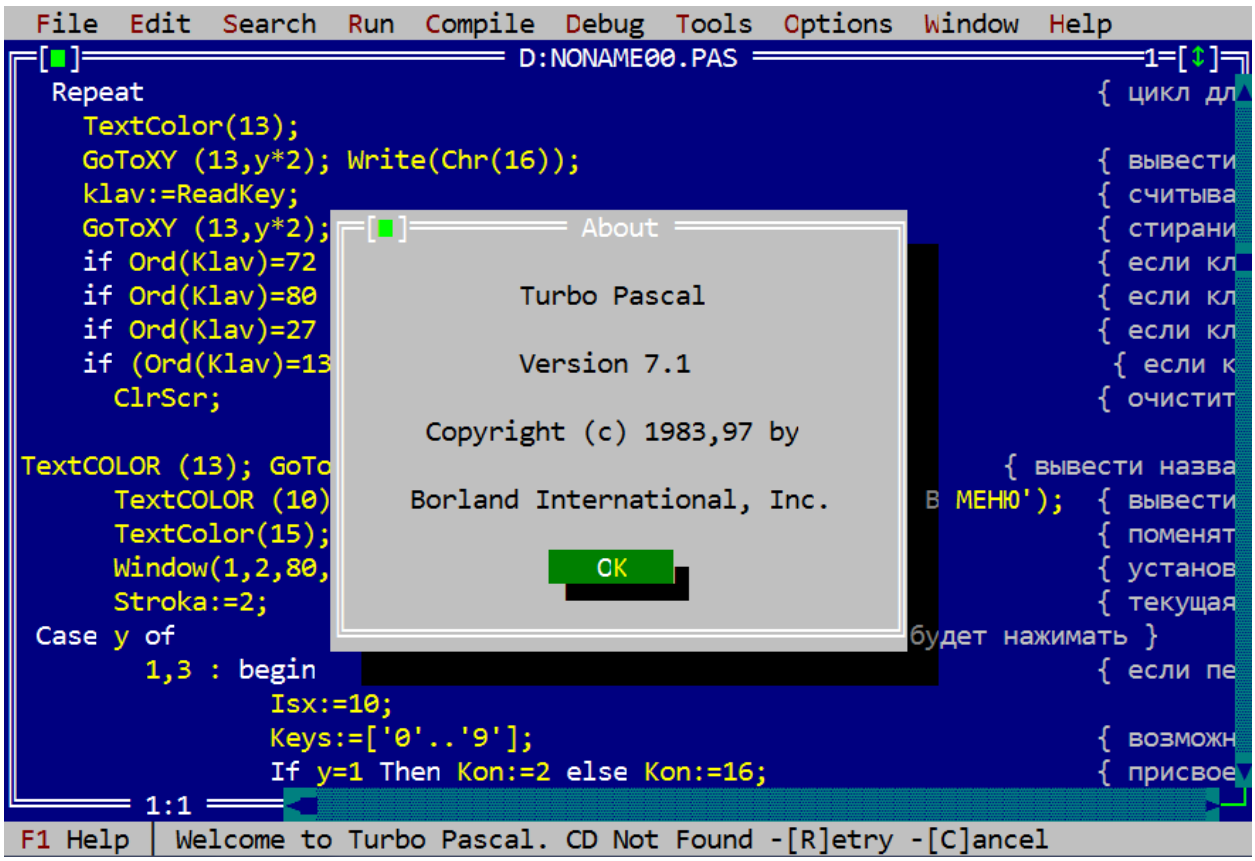

Рисунок 2. Turbo Pascal

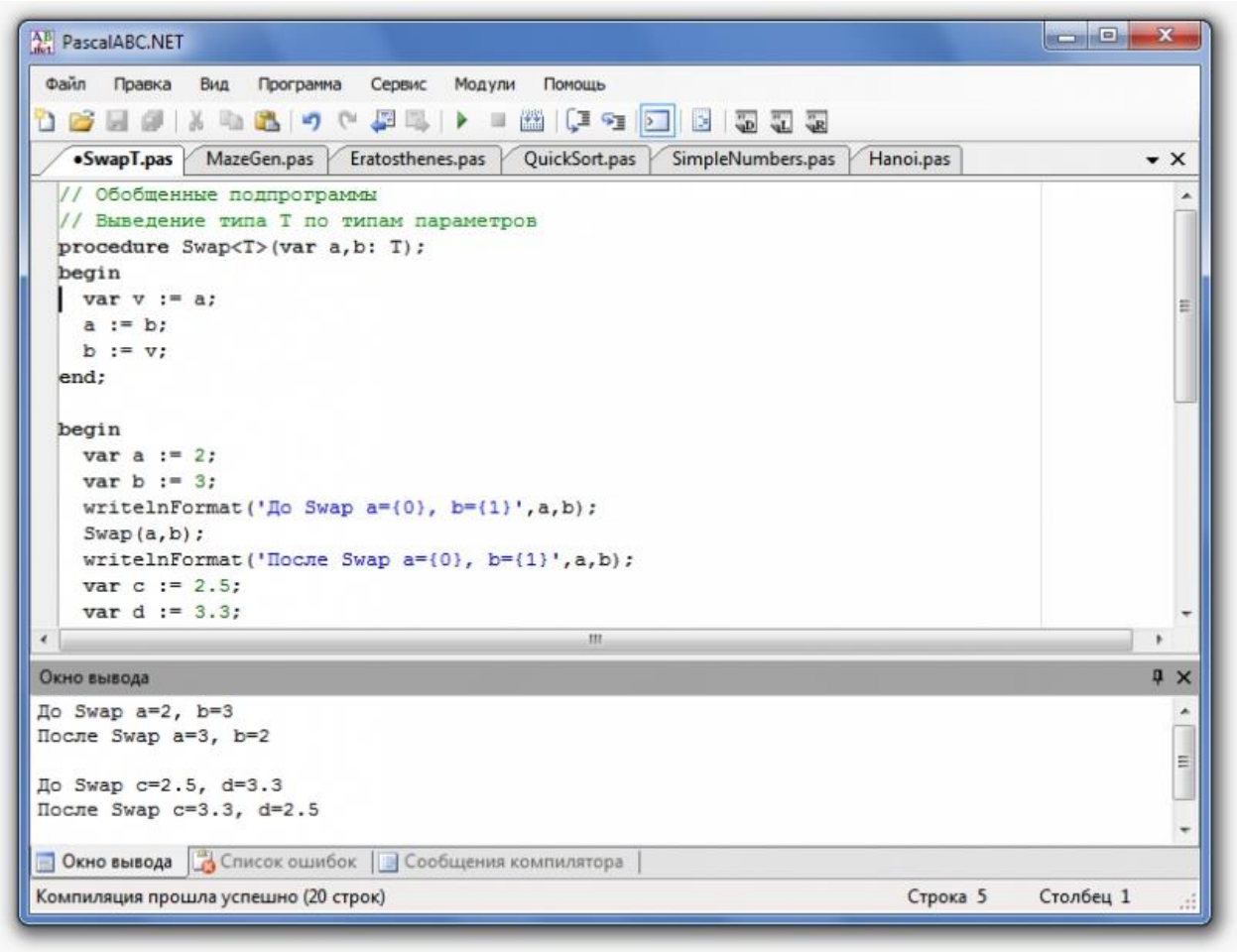


Рисунок 3. PascalABC

**Анализ:** оба интерфейса обладают небольшим числом элементов GUI, фокус разработки - в редакторе кода, деление пунктов меню ясное и чёткое. Отладчик доступен сразу, число опций мало. Интерфейс не меняется от версии к версии.

**Шаг 2:** минимальный работающий пример.

Программа минимального размера, работающая как калькулятор, приведена на врезке [24]:

```
program hello;
begin
    writeln(2, ‘ + ‘, 2, ‘ = ‘, 2 + 2)
```

end.

Анализ: язык опирается на лексемы английского языка, включая оператор “;” (обратите внимание, что в первой программе этого оператора нет!). Используется традиционная запись математических выражений с общепринятым приоритетом операций.

**Шаг 3:** переменные и операторы.

```
var
  x1,x2,x3,y1,y2,y3: boolean;
  kol: byte;
begin
 kol:=0;
 for x1:=false to true do
  for x2:=false to true do
   for x3:=false to true do
    for y1:=false to true do
     for y2:=false to true do
      for y3:=false to true do
       if (((not x1 or x2) and (not x2 or x3)) = true)
          and
          (((not y1 or y2) and (not y2 or y3)) = true)
          and
          ((x1 and y3) = false)
          then
           kol:=kol+1;
 writeln('Количество решений = ', kol);
end.
```

Рисунок 4. Пример переменных и операторов Pascal в программе

**Анализ:** секции кода и данных разделены явно. Название секций (Var, Const) являются комментирующими. Синтаксис описания массивов и переменных близок к речевому (array of, set of, …).

**Шаг 4:** взаимодействие с аппаратной частью ПК (внешним миром).

Pascal поступает с готовой библиотекой для рисования основных графических примитивов: точки, линии, окружности, прямоугольника. Это позволяет обучающимся реализовать свой творческий потенциал практически сразу.

**Анализ:** сторонние библиотеки для работы с графикой, клавиатурой, мышью не требуются.

**Дополнение:** этот приём использован Ильёй Рудольфовичем Дединским в своём авторском курсе (библиотека ТХ, сокращение от “тупой художник” [37,38]).

Все эти достоинства облегчают первоначальное знакомство с языком, но при накоплении опыта делают процесс программирования трудоёмким - ограничения, дающие чистоту кода, начинают требовать всё больше обслуживания:

1. Громоздкие конструкции языка. Сложная программа, написанная на Pascal, с многократным использованием слов begin и end вместо фигурных скобок и иных конструкций, которые упразднены во многих языках, сказываются не в лучшую сторону. Так программа в «блокноте» становится трудночитаемой.
2. Локальные переменные нельзя определять в том месте, где они начинают использоваться – это приводит к засорению пространства видимости, проблемам с оптимизацией (цикл for).
3. Ситуации с неправильным написанием слов (регистр), например, не begin, а Begin, BEGIN или BeGiN. Это неправильно. Программист пишет идентификаторы в правильном регистре, но чужой код, без соблюдения стиля — прочитать очень трудно, а найти ошибки в нем может оказаться и вовсе невозможным.

4. Использование оператора “;” с условным оператором без скобок крайне неудобно - его поиск и удаление при добавлении ветки else практически ни в какой бесплатной IDE штатными средствами не автоматизированы.

Да, решения всех этих сложностей существуют, частично стилевые, частично средствами среды разработки, но это именно решения существующей проблемы - язык навязывает достаточно много непопулярных проектировочных решений.

### 1.3. Анализ основных методических сложностей при изучении языка Си/Си++ как первого языка программирования в старшей школе

Люди стремятся к совершенству во всем, программисты не исключение. С появлением первых языков развитие не остановилось, даже в наше время появляются новые языки программирования, по тем или иным признакам зарабатывающие популярность у разных программистов, зачастую даже для каждой задачи можно выбрать свой язык, например:

- Для программирования приложений под ПК – Delphi, Си, Си++, Visual Basic, Java.
- Для разработки операционных систем, работы с аппаратной частью – Си, Си++, Assembler.
- Для создания кроссплатформенного софта – Java, Python, Ruby.
- Нужен быстродействующий код – Си/Си++.

Появившийся в 1972 году язык Си, разработанный Деннисом Ритчи и Кеном Томпсоном, а позднее его потомок Си++, авторства Бьерна Страуструпа, обладают большой мощью, гибкостью и эффективностью, а потому по сей день сохраняют места в первой пятерке по популярности.

Язык Си оказался привлекателен тем, что он был задуман, как связующее звено между языками высокого и низкого уровня. Таким образом,

он сочетает в себе близкую производительность к языкам низкого уровня (прим.,Ассемблер) и возможность переноса программ между компьютерными платформами, что характерно для языков высокого уровня.

Язык Си был написан программистом-практиком Деннисом Ритчи. Фактически, язык не добавляет лишних конструкций для обслуживания проектировочных амбиций создателя языка - скорее, это язык разметки блока памяти, содержащего исполняемый код и данные для машины фон Неймана. Хорошей иллюстрацией этого тезиса может служить “устройство Даффа” [39]:

```
strcpy(to, from, count)
register char *to, *from;
register count;
{
    register n = (count + 7) / 8;
    if (!count) return;
    switch (count % 8) {
    case 0: do { *to++ = *from++;
    case 7:     *to++ = *from++;
    case 6:     *to++ = *from++;
    case 5:     *to++ = *from++;
    case 4:     *to++ = *from++;
    case 3:     *to++ = *from++;
    case 2:     *to++ = *from++;
    case 1:     *to++ = *from++;
            } while (--n > 0);
    }
}
```

Эрик Раймонд проанализировал причину популярности языка Си с точки зрения того, что интерфейсы должны быть тонкими, без внутреннего состояния и прозрачные, в своей классической работе [40]. Таким образом, популярность языка обусловлена несколькими факторами:

- он занял нишу ассемблера, не привязанного синтаксически к конкретной аппаратной платформе, чем “освободил” огромное количество общих алгоритмов, сделав их кросс-платформенными;
- язык позволяет писать тот код, который требуется разработчикам драйверов и низкоуровневых решений;
- язык распространился в среде юникс-программистов, став неотделимым от их необычной культуры, после чего популярность юникса и линукса катапультировала его на вершину языков для разработки системного ПО;
- аккуратный выбор абстракций, сделанный в юникс: (файловый) поток, сокеты, строки, и написанные для работы с ними стандартные библиотеки фактически создали в языке инструментарий более высокого, чем ассемблер, порядка.

Давайте разберём Си с точки зрения достоинств языка Pascal, перечисленных в предыдущем пункте.

**Шаг 1:** единая среда разработки отсутствует. Под ОС семейства lunix используют компиляторы, вызываемые в командной строке [] и несколько десятков редакторов и IDE: CodeBlocks, CLion, KDevelop, emacs, …

Под ОС Windows абсолютный лидер - Microsoft Visual Studio. Пакет меняется каждые несколько лет, старые версии пакета официально перестают поддерживаться и скачать с официального сайта их невозможно (на 2018 год невозможно найти официальную версию VS2008 community edition). Ситуацию немного улучшают кроссплатформенные решения (CLion,

CodeBlocks, …), но под ОС Windows им требуется установка и настройка компилятора, компоновщика и отладчика, предоставляемых, как правило, третьими лицами.

**Анализ:** IDE рассчитаны на работу с проектами большого размера (исходный код браузера Google Chrome превышает 1.5Gb в архиве, размер исходного кода на июнь 2018 года: 18.8 миллионов строк кода, 3.4 миллиона строк комментариев на трёх основных и 32 дополнительных языках) [41]. Размер настолько велик, что с запасом превышает когнитивные способности людей, которые просто вынуждены полагаться на IDE просто для перемещения и поиска по исходным кодам.

Интерфейс IDE содержит заметно больше элементов GUI и несколько пространств, не связанных с редактированием кода (менеджмент файлов проекта, консоли, ресурсы программы, …).

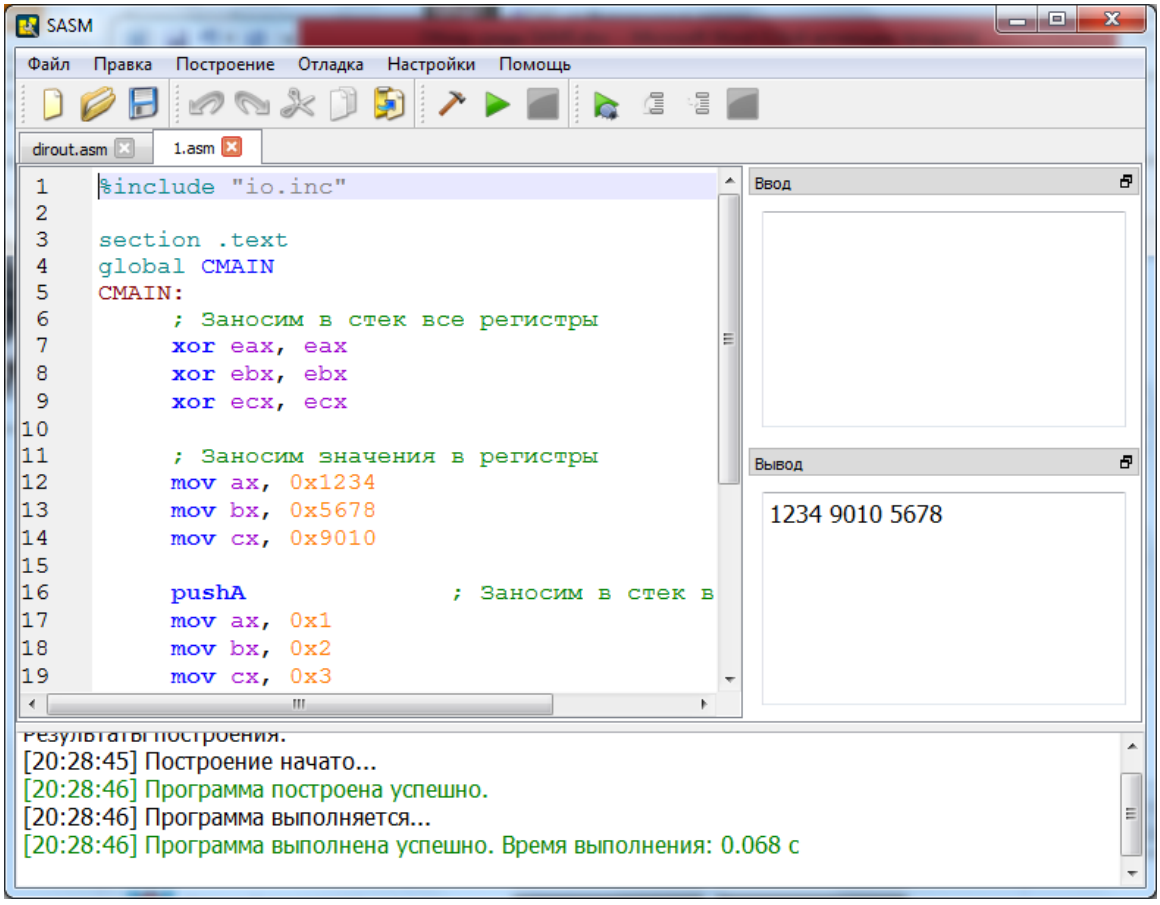


Рисунок 5. Интерфейс SimleASM

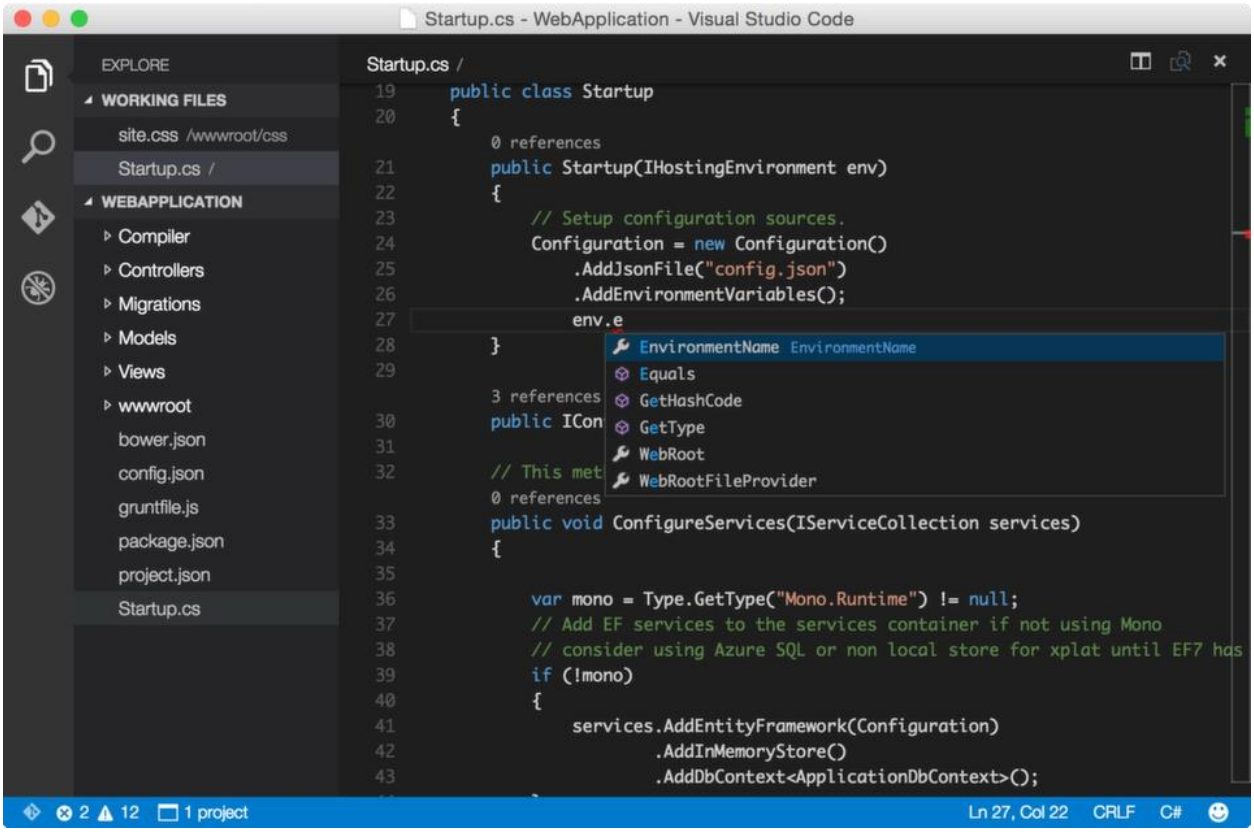


Рисунок 6. Интерфейс VisualStudio

**Вывод:** сильная фрагментация и сопутствующий разброс мнений в инфосфере затрудняют выбор среды для разработки и делают заметную часть материалов для новичков платформо-зависимыми.

**Шаг 2:** минимальный работающий пример.

Сложности, описываемые в данном пункте, детально и независимо разобраны в работе Столярова [31] и красноречиво называется *“Hello, world, или барьер, который возьмут не все”*.

Программа минимального размера, аналогичная Паскалевской, приведена ниже (пример сделан на Visual Studio 2008, который ещё можно встретить в учебных заведениях) может быть сделана только повторением за учителем, поскольку требуется создать новый проект, тип которого (Console Application) надо выбрать из 5 - 40 возможных. Её текст:

```
#include "stdafx.h"                                    // 1
```

```
#include <math.h>                          // 2
int _tmain(int argc, _TCHAR* argv[])       // 3, 4, 5, 6, 7
{
    printf("%d + %d = %d\n", 2, 2, 2 + 2); // 8, 9
    return 0;                              // 10
}
```

**Анализ:** программа содержит 10 ссылок вперёд на материал, который будет изучен только в течение минимум семестра (макропроцессор часто вообще не изучается):

| | |
|---|---|
| 1 | Макро-директива включения локального файла, неявно созданного студией.<br>Нестандартна. Ученики после этого не умеют подключать стандартные библиотеки. |
| 2 | Математики из коробки нет, требуется ручное подключение. |
| 3 | Сразу вводится тип int и понятие возвращаемого типа функции и типа переменной. |
| 4 | _TCHAR - это нестандартный макрос, используемый фирмой Микрософт для локализации программ на Си. |
| 5 | Указатель. Одна из самых проблемных тем. |
| 6 | Тип массив. |
| 7 | Функции main просто нет. |
| 8 | Язык описания форматов функции printf. Отдельная лекция. |
| 9 | Специальный символ ‘\n’. |
| 10 | Оператор возврата значения из функции. |

**Вывод:** нарушены большинство дидактических принципов.

**Шаг 3:** переменные и операторы.

Зоны видимости уже делятся на локальные и глобальные. Возможность определять переменные внутри операторов (for, начиная с C99; if, начиная с C++98) запутывает картину, усложняя работу в случае списывания примеров с сети - ученики просто не понимают, что они списали.

**Вывод:** полученная выразительность в ходе профессиональной деятельности позволяет решать многие проблемы языка Pascal, но на ранних этапах обучения учителю следует особенно осторожно выстраивать курс.

**Шаг 4:** взаимодействие с аппаратной частью ПК (внешним миром).

Есть только на уровне библиотек третьих лиц, тесно интегрировано с операционной системой хоста, платформенно-зависимо, требует понимания архитектуры ОС и архитектуры устройства.

**Анализ:** крайне чувствительно ко всем перечисленным выше факторам. Даже после выдачи видеодемонстрации процесса установки, из 17 студентов физфака КГПУ справилось трое.

Обнаруженные ошибки:

- несоответствие версии компилятора;
- несоответствие разрядности ОС;
- автоматическая настройка путей до компилятора, идущего с другой версией IDE;
- неправильная настройка выбора между статической и динамической компоновкой, вызванная разногласием между видео-учебниками на youtube;
- ошибка, вызванная нестандартным символом в пути файла.

Разрабатываемая платформа и предназначена для решения перечисленных трудностей.

Несколько слов о языке Си++ – это язык программирования высокого уровня с поддержкой нескольких парадигм программирования: объектно-ориентированной, обобщенной и процедурной. Разработан Бьёрном Страуструпом в 1979 году на базе языка Си.

При создании Си++ стремились сохранить совместимость с языком Си. Большинство программ на Си будут исправно работать и с компилятором Си++.

Си++ имеет следующие нововведения по сравнению с Си:

- поддержка объектно-ориентированного программирования через классы;
- поддержка обобщенного программирования через шаблоны;
- дополнение к стандартной библиотеки;
- составные типы данных;
- обработка исключений;
- пространства имен;
- встроенные функции;
- перегрузка операторов;
- перегрузка имен функций;
- ссылки и операторы управления свободно распределенной памятью.

Требования совместимости с Си на уровне синтаксиса и семантики, осложнённое условием нулевого штрафа на производительность старого кода привело к тому, что сегодня это один из самых комплексных и динамически развивающихся языков в мире. Каждый год собирается конференция CppCon только для того, чтобы люди успевали ознакомиться с новыми возможностями и осознать, что можно изменить в старом коде. Полное

освоение Си++, если возможно, требует нескольких лет полноценной работы (40 часов в неделю) под руководством ментора в среде уже обученных разработчиков и чтения отобранных ментором книг и других материалов.

Выдержка про Си++: название «Си++» была придумана Риком Масситти и впервые использована в декабре 1983 года. Ранее, на этапе разработки, новый язык назывался «Си с классами». Имя, получилось в итоге, происходит от оператора Си «++» (увеличение значения переменной на единицу) и распространенном способа присвоения новых имен компьютерным программам, заключается в добавлении к имени символа «+» для обозначения улучшений. Согласно словам автора языка, «это название указывает на эволюционную природу изменений Сi».

В одной из книг [25] Б.Страуструп описывает принципы, которых он придерживался при проектировании Си++. Эти принципы объясняют, почему Си++ именно такой, какой он есть. Некоторые из них:

1. Получить универсальный язык со статическими типами данных, эффективностью и переносимостью языка Си.
2. Непосредственно и всесторонне поддерживать множество стилей программирования, в том числе процедурное программирование, абстракцию данных, объектно-ориентированное программирование и обобщённое программирование.
3. Дать программисту свободу выбора, даже если это даст ему возможность выбирать неправильно.
4. Максимально сохранить совместимость с Си, тем самым делая возможным лёгкий переход от программирования на Си.
5. Избежать разночтений между Си и Си++: любая конструкция, допустимая в обоих языках, должна в каждом из них обозначать

одно и то же и приводить к одному и тому же поведению программы.

6. Избегать особенностей, которые зависят от платформы или не являются универсальными.
7. «Не платить за то, что не используется» — никакое языковое средство не должно приводить к снижению производительности программ, не использующих его.
8. Не требовать слишком усложнённой среды программирования.

Сторонники Си++ считают, что в сравнение с другими языками благодаря этой особенности у него широта возможностей больше. Создатель языка и его последователи считают, что *«стандарт — это контракт между программистами, разрабатывающими программы на языке, и программистами, разрабатывающими компиляторы языка»*.[26]

Си++ работает под управлением многих операционных систем. К ним относятся: Mac, Linux (включая дистрибутив Ubuntu всех версий), Windows.

Язык Си позволяет программистам максимально контролировать программу, а Си++ идет по пути усложнения компилятора, чтобы позволить программисту писать программу так, как ему «удобно». Сейчас снова же, в идеальном случае, компилятор языка поймет желание программиста и полученный код будет, все-таки, максимально эффективным (или близким к эффективному). Поэтому подход Си++ не может не вызвать интереса и одобрения, поскольку дает возможность создания эффективных программ, не снижая при этом их читабельность или удобство наращивания.

Си++ уже стал универсальным языком для программистов всего мира, языком, на котором написано следующее поколение высокоэффективного программного обеспечения; большинство современных операционных систем Windows, Unix, DOS написаны именно на этом языке (или на ее

разновидности). По мнению автора языка, различия между идеологией Си и С++ заключается примерно в следующем: программа на Си отражает «образ мышления» процессора, а Си++ – способ мышления программиста.

В 1990-х годах Си++ стал одним из важнейших языков программирования общего назначения. Язык используют для системного программирования, разработки программного обеспечения, написание драйверов, мощных серверных и клиентских программ, а также для разработки развлекательных программ, таких как видеоигры. Си++ существенно повлияла на другие, популярные сегодня, языки программирования: C# и Java. В настоящее время считается господствующим языком, используемым для разработки коммерческих продуктов, 90% игр пишутся на нем.

### 1.4. Проектирование системы заданий на основе принципов последовательности, систематичности, наглядности

К сожалению, языки программирования проникают в школу заметно медленнее, чем остальное программное обеспечение. Осознав всю необходимость, продиктованную временем, перед учителем встает вопрос, какую же конкретный язык программирования использовать на уроках?

Но здесь он сталкивается с другими проблемами. Отсутствие апробированных методик, уровень входа языка, доступность для понимания и другие.

По Си++ существует огромное количество учебной литературы, переведённой на всевозможные языки. Но это все равно язык с высоким порогом вхождения, хотя и среди всех языков такого рода обладает наиболее широкими возможностями. Из-за сложившейся специфики обучения Си/Си++ требуется огромное мастерство и опыт для написания программ, так как «небрежные» коды с высокой вероятностью могут оказаться некомпилируемыми.

Но научить программировать можно, только программируя, решая конкретные задачи. А Си/Си++, как и большинство других языков программирования, разрабатывались не для обучения, а как средство профессиональной деятельности.

И снова обращаемся к современным тенденциям. В последнее время появляются различные среды быстрого проектирования на Си++, которые предоставляют готовые компоненты для ввода и вывода информации и поддерживают концепцию виртуального проектирования. Они дают возможность подросткам испытать себя в роли программиста. При этом написание кода проходит в столь им близкой – игровой форме и результат своей творческий игры ученики получают сразу же.

Яркий пример такому подходу служит игра, использованная в «Часе кода», где учащийся должен собрать из пазлов с прописанными элементами кода программу, которая проведет персонажа по извилистому лабиринту. Подобный подход значительно упрощает изучение языка, но не делает его полноценным. Так как хоть и присутствует последовательность в изучении материала, обучающийся не получит навыков работы со средой (платформой). К тому же спектр задач чётко прописан.

Таким образом, создание среды, в которой изучение языка будет систематическим, последовательным (от простых действий и конструкций к более сложным) и наглядным, позволит сделать его более доступным и привлекательным для изучения его в старших классах, как основного языка программирования. К тому же это понизит порог входа и может значительно повысить уровень усвоения материала.

Задания в графической среде могут быть разнообразными, но главное, что они должны соответствовать принципам последовательности, систематичности, наглядности. То есть мы не можем делать так:

1. Объект должен пройти лабиринт за 7 действий;

2. Посчитать 2+2;

3. Посчитать √451;

4. Сдвинуть объект на одну клетку в верх.

Каждое следующее задание должно вытекать из предыдущего, усложняя его и давая возможность использовать уже знакомый элемент код, а также опробовать новый. Наш процесс обучению должен быть похож на реку. Она начинается с истока, маленького ключа, что бьет из-под земли, по мере течения, в ручей вливаются другие ручейки и это уже речка, что с каждым новым притоком все шире и сильнее, а в конце эта река впадает в море. Наши учащиеся открывая новые возможности в песочнице, что мы им дали, получая новые знания и навыки, постепенно открывают для себя не только язык, но и среду с ее возможностями, а изучив и ее, они уже могут создать полноценную программу.

Для мобильности, задания рекомендуем писать в шапке среды, в которой будет работать учащийся на уроке. Там же можно давать дополнения, пояснения, подсказки. Таким образом не только условия задания будут перед глазами у ученика, но и рабочие элементы, с которыми он сегодня знакомится.

В качестве заданий я предлагаю рассмотреть ряд задач на перемещение нашего зеленого «колобка».

**Задача 1.** Передвиньте «колобок» на две клетки вправо.

Для написания программы, вы можете использовать следующие конструкции:

```
me.go_left();
me.go_right();
me.go_up();
me.go_down();
```

**Задача 2.** Передвиньте «колобок» на три клетки левее и ниже.

**Задача 3.** Проведите «колобок» по кругу радиусом 6 клеток 4 раза.

Для написания программы, вы можете использовать, ранее изученные конструкции, а также конструкции цикла. Пример, написания цикла:

```
int i; /* Цикл будет работать до тех пор, пока i < 10, при этом после каждой итерации переменная i будет инкрементироваться(увеличиваться на 1)*/

    for ( i = 0; i < 10; i++ ) { /* Имейте ввиду что условие проверяется перед каждым повторением, то есть работа цикла остановится когда переменная i будет равна 10*/

        printf( "%d\n", i );

    }
```

**Задача 4.** «Колобок» должен сместиться в правый угол на 6 клеток и посчитать количество шагов.

# ГЛАВА 2. Среда для изучения языков программирования Си/Си++

## 2.1. Выбор платформы разработки и предварительная настройка

Прежде, чем приступить к созданию двумерной графической среды, необходимо выбрать среду исполнения (IDE, далее, - платформа), в которой будет писаться и компилироваться наша программа.

Таким образом, платформа должна соответствовать следующим критериям:

1. работает с языками Си и Си++;
2. кросплатформенна;
3. доступна;
4. проста в использовании;
5. не требовать дополнительных установок.

Кросплатформенность позволит нашей среде работать на любой ОС, что актуально, так как в одном образовательном учреждение работают на Linux, в другом на Windows 7, в третьем могут и вовсе ещё работать с XP. А мы стремимся к тому, чтобы наша среды была как можно доступнее и работать с ней могли повсеместно.

Платформа не должна требовать каких-либо разрешений к использованию и, тем более, финансовых затрат, то есть быть абсолютно доступной и юридически чистой.

Простота платформы позволяет быстрее и комфортнее работать над программой как при её создании, так и непосредственно при обучении учащимся и учителям.

А отсутствие необходимости устанавливать компоненты делает нашу платформу переносимой, что позволит с легкостью запускать её на разных ПК.

Наш выбор пал на IDE Code::Blocks[42], с которой мы уже работали во время обучения “Языкам и методам программирования” в КГПУ им. В.П. Астафьева. Эта IDE – свободная, кроссплатформенная платформа для разработки, написанная на Си++ и работающая с Си и Си++. У неё открытая архитектура, можно масштабировать окно разработчика, что позволит сделать размер строк оптимальным для каждого пользователя. Code::Blocks может работать на ОС Windows, Linux и Mac OS X.

Платформа имеет широкие возможности компиляции, поддерживая множество компиляторов, может работать с многопрофильными проектами.

Кроме того, в Code::Blocks удобный интерфейс, включающий подсветку синтаксиса, возможность сворачивать блоки кода, имеется автодополнение кода, есть браузер классов, присутствует поиск по проекту с подсветкой найденных совпадений, и множество иных дополнительных возможностей.

Компилятор ставится независимо от среды, поэтому можно отдельно обновлять любую из этих компонент. Для Code::Block существует множество видеоуроков и курсов. [43,44,45] Развитый набор опций командной строки позволяет гибко управлять процессом запуска оболочки.

Для обеспечения независимости от привилегий администратора, вся дальнейшая работа строится в portable версии IDE, которая идёт в комплекте со всеми расширениями и компилятором gcc в версии для Windows, разрабатываемого и поддерживаемого в рамках проекта MinGW [46]

### 2.2. Настройка структуры проекта для работы без функции main и внешних библиотек

Большинство нетривиальных элементов, делающих невозможным мягкое знакомство с языком Си, связаны с архитектурой операционной системы: вызов функции main порождающим процессом, возвращаемое

значение, использование указателей. Так как Code::Blocks позволяет создавать проекты и открывать их использованием опции командной строки, сложность создания проекта можно обойти, сделав заготовки на все задания и сохранив их. Для запуска среды для работы над конкретным проектом достаточно распаковать архив с готовой версией среды и дважды щёлкнуть по командному файлу, внутри которого расположен тривиальный скрипт:

```
@cd CodeBlocks_16.01
CbLauncher.exe --no-check-associations ..\Lab_projects\lab_01\lab_01.cbp
```

Первая команда сменяет рабочий каталог, вторая - запускает IDE и открывает файл проекта. Использование только относительных путей делает код переносимым.

В проекте создано два файла, main.cpp имеет вид:

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    // Точка входа команд пользователя.
    // Фигурные скобки экранируют локальные переменные от кода юзера.
    {
    #include "user.cpp"
    }
```

```
    cout << "Programma zakonchilas'\n";
    return 0;
}
```

Фактически, все сложные части программы остались в этом файле, который включает в себя с помощью директивы макропроцессора другой файл проекта, user.cpp.

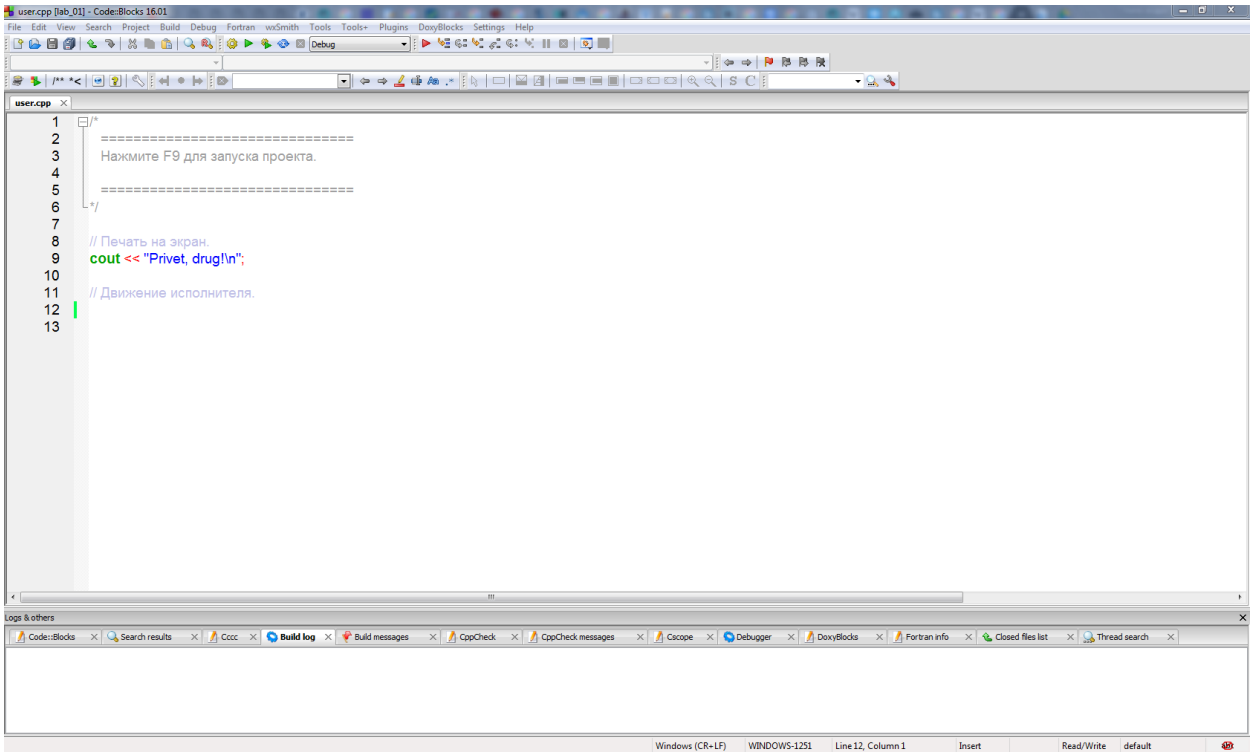


Рисунок 7. Начальный экран

Именно его и видит обучающийся после запуска проекта. Поскольку содержимое этого файла внедряется внутрь функции main, сам фрагмент, повторяющий вариант программы на Pascal, выглядит так:

```
cout << 2 << “ + “ << 2 << “ = “ << 2 + 2 << endl;
```

Одна строка кода и всё.

Первоначальная цель достигнута - мы можем создавать переменные (они все будут локальными), делать циклы, ветвления и так далее.

### 2.3. Графический исполнитель: структура и архитектура приложения

Использование графического исполнителя показало себя как неоценимый педагогический приём (Кумир в отечественной школе [47], Karel в курсе информатики Стенфорда [48]). Для дальнейшей работы над средой, нам необходимо настроить классы, которые позволят создавать графические объекты. Для этого нужно их подключить. Оптимальный вариант - использовать готовые библиотеки классов. Для моего проекта была выбрана Simple and Fast Multimedia Library (SFML), которую можно легко получить из открытого доступ на сайте https://www.sfml-dev.org.

Шаг 1. Здесь необходимо выбрать версию для ОС ПК, на которых будет запускаться программа. Дополнительной установки SFML не требует, остается только интегрировать ее в нашу графическую среду.

Шаг 2. Запустив проект отправляемся в настройки сборки проекта. В открывшемся окне, выбираем вкладку Search directories - Compiler. Выбираем нашу среду и добавляем новую папку, которая будет находиться в том же месте, где и наша среда.

Шаг 3. Достаем библиотеку из архива и переносим все содержимое в созданную ранее папку.

Шаг 4. Вновь открываем Code::Blocks и указываем путь для поиска заголовочных файлов. Для этого в окне, где создавали папку, прописываем “.\include”. Это позволит скомпилировать нашу среду на любом ПК, куда ее перенесут.

Шаг 5. В Search Directories  переходим во вкладку Linker и добавляем дополнительный путь в библиотеку: “.\lib”.

Шаг 6. В левом окошке выбираем Release и принимаем внесенные изменения.

Шаг 7. В Release открываем вкладку Linker setting. Сюда мы добавляем библиотеки, которые необходимо подключить: “sfml-system”, “sfml-window”, “sfml-graphics”. Для того, чтобы библиотеки были подключены и в нашей релизной версии и отладочной (на случай багов). Выделяем все три библиотеки и копируем их в Debug. Заходим в отладочную версию и добавляем “-d”, чтобы наши библиотеки заработали.

Теперь библиотека классов подключена, и мы имеем переносимую версию проекта с настроенной графической библиотекой и можем переходить к следующему пункту – созданию графического исполнителя.

Исполнитель делается как объект класса Leo, методы которого пользователь может использовать для воздействия на исполнителя. Это решение имеет ряд достоинств:

- благодаря инкапсуляции у нас есть полный контроль над методами объекта и его состоянием;
- автодополнение работает как контекстная справка по доступным методам.

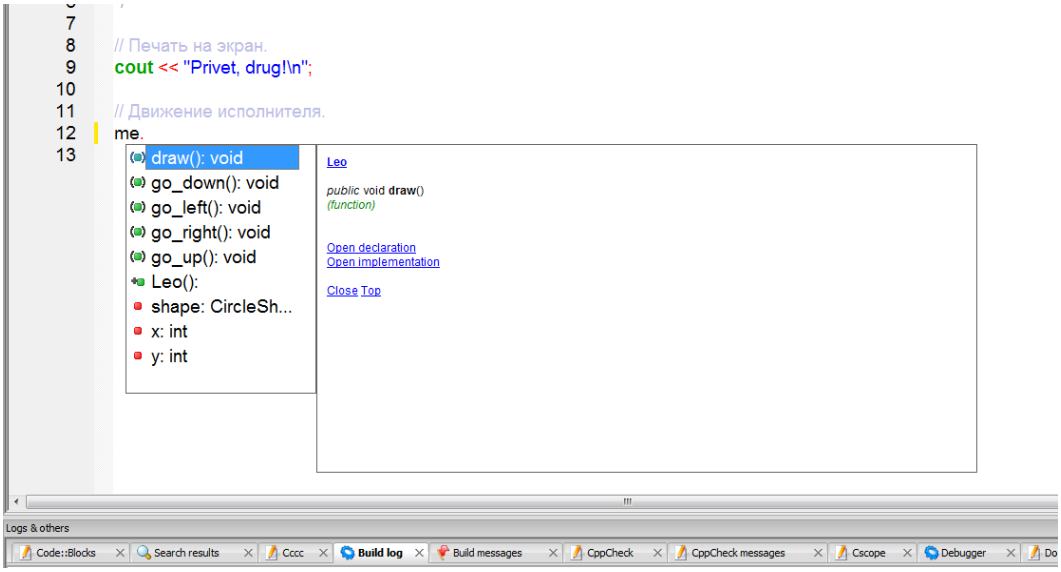


Рисунок 8. Автодополнение

Таким образом, мы получаем возможность опереться на все методические наработки, созданные в отечественной школе преподавания на базе платформы “Кумир”. Следует отметить, что после выхода из пользовательского фрагмента кода, мы получаем возможность исследовать состояние исполнителя с помощью кода, размещённого в теле функции main, что позволит нам выполнить дополнительный контроль выполнения, поставленного перед обучающимся задания, если это потребуется.

## ЗАКЛЮЧЕНИЕ

В ходе работы над ВКР была разработана двумерная графическая среда для обучения программированию на языке Си/Си++ учащихся старшей школы. Данная среда является мобильной и кросплатформенной, она может работать на любой операционной системе и не требует установки и дополнительных настроек. Достаточно перенести на флэшке или скачать на компьютер папку с программой и запустить ее.

С помощью разработанной среды обучение Си/Си++ не только упрощается и становится более доступным, но и позволяет построить его в соответствии с дидактическими принципами систематичности и последовательности, а визуальный исполнитель делает сам процесс программирования для учащегося наглядным.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК


1. Абрамов В. Г., Трифонов Н. П., Трифонова Г. Н. Введение в язык Pascal/Учеб. пособие. – 1988.
2. Артемов В. А. Психология наглядности при обучении //М.: Просвещение. – 2004.
3. Бондарев В. М., Рублинецкий В. И., Качко Е. Г. Основы программирования. – Харьков; Ростов н/Д : Фолио: Феникс, 1997.
4. Бородин М. Н. Программы для общеобразовательных учреждений: Информатика. 2-11 классы //БИНОМ. Лаборатория знаний. – 2007.
5. Вандевурд Д. Шаблоны C++: справ. разработчика:[пер. с англ.]. – Издательский дом Вильямс, 2008.
6. Вирт Н. Pascal: Руководство пользователя и описание языка/К. Йенсен, Н. Вирт //М.: Компьютер. – 1995.
7. Грогоно П. Программирование на языке Pascal. – 1982.
8. Дейтел Х. М., Дейтел П. Д. Как программировать на C++. – 2005.
9. Дж В. 6. Применение шаблонов проектирования. Дополнительные штрихи //М.: Вильямс. – 2003.
10. Епанешников А. М. и др. Программирование в среде TURBO PASCAL 7.0. – М. : АО" Диалог-МИФИ", 1995.
11. Зубов В. С. Программирование на языке Turbo Pascal (версии 6.0 и 7.0). – М. : Информ.-изд. дом" Филинъ", 1997.
12. Карпов Б., Баранова Т. C++. – Питер, 2005.
13. Керниган Б. В. Язык программирования C, 2-е издание. – Издательский дом Вильямс, 2012.
14. Кузнецов А. А. и др. Непрерывный курс информатики (концепция, система модулей, типовая программа) старшая школа (X-XI классы) //Информатика и образование. – 2005. – №. 6. – С. 2-14.

15. Кушниренко А. Г., Лебедев Г. В. Информатика: 12 лекций о том, для чего нужен школьный курс информатики и как его преподавать //Лаборатория Базовых Знаний. – 2000.
16. Либерти Д. Освой самостоятельно C++ за 21 день:[пер. с англ.]. – Издательский дом Вильямс, 2008.
17. Липпман С. Б., Лажойе Ж., Му Б. Язык программирования C++. – Вильямс, 2007.
18. Перминов О. Н. Программирование на языке Pascal. – Радио и связь, 1988.
19. Подласый И. П. Педагогика: 100 вопросов–100 ответов: Учеб. пособие для студ. высш. учеб. заведений //М.: Изд-во Владос-пресс. – 2001.
20. Реймонд Э. С. Искусство программирования для Unix. - Издательский дом Вильямс, 2005.
21. Семакин И. Г., Хеннер Е. К. Информатика и ИКТ. Базовый уровень. 10-11 классы: методическое пособие //М.: Бином. Лаборатория знаний. – 2011.
22. Семакин И. Г., Хеннер Е. К. Информатика и ИКТ. Базовый уровень: учебник для 10-11 классов //М.: Бином. Лаборатория знаний. – 2009. – Т. 246. – С. 6.
23. Скрипкин К. Г. Экономическая эффективность информационных систем М.: ДМК Пресс. 256 с.
24. Столяров А. В. Программирование: введение в профессию. 1: Азы программирования //М.: МАКС Пресс. – 2016.
25. Страуструп Б. Дизайн и эволюция C++. – 2006.
26. Страуструп Б. Программирование. Принципы и практика использования C++. – Litres, 2017.
27. Сухомлин В. А. Введение в анализ информационных технологий. – Горячая линия-Телеком, 2003.

28. Угринович Н. Д. Информатика и ИКТ. Базовый уровень: учебник для 11 класса //М.: Бином. Лаборатория знаний. – 2008. – Т. 2010.
29. Угринович Н. Д. Преподавание курса" Информатика и ИКТ" в основной и старшей школе. – 2005.
30. Эккель Б., Эллисон Ч. Философия C++. – Питер, 2004.
31. Столяров А. В. Эссе "Язык Си и начальное обучение программированию" [Электронный ресурс] //Режим доступа: http://www.stolyarov.info/pvt/anti_c
32. Студия Кода. Каталог курсов [Электронный ресурс] //Режим доступа: https://studio.code.org
33. Час Кода [Электронный ресурс] //Режим доступа: http://www.coderussia.ru/
34. CodeCombat [Электронный ресурс] //Режим доступа: https://codecombat.com/
35. Бикмеев А. Подключение библиотеки SFML в IDE Code::Blocks [Электронный ресурс] //Режим доступа: https://www.youtube.com/watch?v=_Efc_bOUQUY
36. Олимпиада НТИ. Интеллектуальные робототехнические системы. [Электронный ресурс] //Режим доступа: http://nti-contest.ru/participants/
37. Дединский И. Р. Библиотека Тупого Художника [Электронный ресурс] //Режим доступа: http://ded32.net.ru/load/1-1-0-4
38. Дединский И. Р. Библиотека Тупого Художника [Электронный ресурс] //Режим доступа: http://storage.ded32.net.ru/Lib/TX/TXUpdate/Doc/HTML.ru/
39. ВикипедиЯ. Свободная энциклопедия. Устройство Даффа [Электронный ресурс] //Режим доступа: https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_%D0%94%D0%B0%D1%84%D1%84%D0%B0

40. Раймонд Э. С. Искусство программирования Unix [Электронный ресурс] //Режим доступа: http://www.catb.org/esr/writings/taoup/html/ch04s03.html#c_thin_glue
41. Chromium (Google Chrome) [Электронный ресурс] //Режим доступа: https://www.openhub.net/p/chrome
42. Code::Blocks [Электронный ресурс] //Режим доступа: http://www.codeblocks.org/
43. Averbah Andrew C++ Урок 0. Установка Code Blocks, создание проекта. [Электронный ресурс] //Режим доступа: https://www.youtube.com/watch?v=iZrtPgMEK8M
44. Thenewboston C Programming Tutorial - 2 - Setting Up Code Blocks [Электронный ресурс] //Режим доступа: https://www.youtube.com/watch?v=3DeLiClDd04
45. Позднеев А. В. Введение в язык C++, среда программирования Code::Blocks, простейшие алгоритмы и программы [Электронный ресурс] //Режим доступа: http://ani.cmc.msu.ru/files/geo-2011-prac-01.pdf
46. Minimalist GNU for Windows [Электронный ресурс] //Режим доступа: http://mingw.org/
47. Система программирования КуМир [Электронный ресурс] //Режим доступа: https://www.niisi.ru/kumir/
48. STANFORD KAREL [Электронный ресурс] //Режим доступа: http://stanford.edu/~cpiech/karel/learn.html
49. TIOBE [Электронный ресурс] //Режим доступа: https://www.tiobe.com/tiobe-index/