МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ЧЕБОКСАРСКИЙ ИНСТИТУТ (ФИЛИАЛ)
МОСКОВСКОГО ПОЛИТЕХНИЧЕСКОГО УНИВЕРСИТЕТА

Кафедра Информационных технологий, электроэнергетики и систем управления

КУРСОВАЯ РАБОТА

по дисциплине: Программирование на языке низкого уровня
на тему: «Разработка игры викторины на языке С»

Выполнила:
студентка группы 09.03.01
Ефимова Александра Владимировна
учебный шифр 19094

Проверил:
Решетников А.В.

Чебоксары 2021

Содержание

Введение

Целью данной курсовой работы является разработка игры викторины на языке С.

Объектом исследования являются игры на языке С.

Предметом исследования является игра викторина на языке С.

На Земле 7 миллиардов уставших от жизни людей, поэтому небольшая консольная игра, несущая в себе развлекательный посыл, всегда будет актуальна, главное, что с ее помощью люди смогут не только развлекаться, но и тренировать мозги. **В этом заключается актуальность курсовой работы**.

В соответствии с целью работы были сформулированы **следующие задачи**:

1. Ознакомиться с основной теорией по низкоуровневому программированию;
2. Изучить, что оно из себя представляет, его характеристики и назначение;
3. Рассмотреть список популярных низкоуровневых языков;
4. Проанализировать, какие имеются преимущества и недостатки языков этого вида;
5. Выбрать средства разработки, выбрать язык программирования для написания разработки, а также среду разработки;
6. Проанализировать технического задание и разработать игру викторину на выбранном языке;
7. Описать результат разработки;
8. На основе выполненной разработки составить выводы о проделанной работе.

В введении была поставлена цель работы, а также сформулированы задачи, была обоснована актуальность, сформированы объект и предмет исследования.

В первой главе содержится информация об основной теории по низкоуровневому программированию, его определение, характеристики и назначение.

Во второй главе объясняется выбор средств разработки, а именно выбор языка программирования и среды разработки.

В третьей главе подробно рассказывается о самой разработке. Здесь находится анализ технического задания, а также описание готового продукта.

Заключение содержит в себе выводы о проделанной работе.

Список источников включает в себя все материалы, которые использовались для работы.

1. Теория по низкоуровневому программированию.

1.1. Понятие, характеристики, назначение.

Низкоуровневым языком программирования называют язык, близкий к программированию непосредственно в машинных кодах эксплуатируемого реального или виртуального процессора.

В данном случае машинные команды обозначают с помощью мнемоники, в форме продуманных сокращений слов человеческого языка (обычно английских), а не в виде ряда из двоичных нулей и единиц. В некоторых случаях одному мнемоническому значению соответствует совокупность машинных команд, предназначенных для выполнения одинакового действия с разными ячейками памяти процессора.

Низкоуровневые языки программирования также могут обладать дополнительными возможностями, в том числе, макроопределения (или макросы). С помощью директив управляют трансляцией машинных кодов с занесением констант и литеральных строк, резервированием памяти для переменных и размещением исполняемого кода по конкретным адресам.

Благодаря этим языкам можно оперировать не конкретными, а переменными ячейками памяти. Работа осуществляется с учетом особенностей конкретного семейства процессоров.

Программы для первых компьютеров писались в двоичных машинных кодах, что представляло собой достаточно трудоемкую и сложную задачу. Упростить процесс позволили низкоуровневые языки программирования. С их помощью машинные команды приобрели более понятный вид. Функцию преобразования их в двоичный код выполняли особые программы (трансляторы) двух видов:

- компиляторы, необходимые для трансформации текста программы в машинный код, сохраняемый и затем используемый уже без компилятора, например файлы с расширением *.exe;

- интерпретаторы, которые преобразуют частично программу в машинный код, выполняют и далее переходят к следующей части.

Программист, специализирующийся на написании алгоритма для компьютера на низкоуровневом языке, обращается напрямую к компьютерным ресурсам:

- процессору;
- памяти;
- периферийным устройствам.

Такой процесс гарантирует высокую скорость функционирования программ, что объясняется отсутствием скрытых фрагментов кода, добавляющих автоматически компилятор во время трансформации исходного кода в бинарный.

При использовании низкоуровневых языков за все ресурсы внутри компьютера, включая время загрузки процессора и выделяемую память, ответственен программист. В связи с этим языки низкого уровня считают небезопасными, что объясняется большим количеством ошибок в программном коде по сравнению с высокоуровневыми языками.

Низкоуровневое программирование используют для разработки компактного программного обеспечения: такого, как системы реального времени; микроконтроллер; драйверы, управляющие внешними устройствами (включая принтеры, сканеры, камеры).

1.2. Список популярных низкоуровневых языков.

Ассемблер (Assembly) — язык программирования, понятия которого отражают архитектуру электронно-вычислительной машины. Язык ассемблера — символьная форма записи машинного кода, использование которого упрощает написание машинных программ [5]. Для одной и той же ЭВМ могут быть разработаны разные языки ассемблера. В отличие от языков высокого уровня абстракции, в котором многие проблемы реализации алгоритмов скрыты от разработчиков, язык ассемблера тесно связан с системой команд микропроцессора. Для идеального микропроцессора, у которого система команд точно соответствует языку программирования, ассемблер вырабатывает по одному машинному коду на каждый оператор языка. На практике для реальных микропроцессоров может потребоваться несколько машинных команд для реализации одного оператора языка.

Язык ассемблера обеспечивает доступ к регистрам, указание методов адресации и описание операций в терминах команд процессора. Язык ассемблера может содержать средства более высокого уровня абстракции: встроенные и определяемые макрокоманды, соответствующие нескольким машинным командам, автоматический выбор команды в зависимости от типов операндов, средства описания структур данных. Главное достоинство языка ассемблера — «приближенность» к процессору, который является основой используемого программистом компьютера, а главным неудобством — слишком мелкое деление типовых операций, которое большинством пользователей воспринимается с трудом. Однако язык ассемблера в значительно большей степени отражает само функционирование компьютера, чем все остальные языки.

На языке ассемблера пишут программы или их фрагменты в тех случаях, когда критически важны:

- объем используемой памяти (программы-загрузчики, встраиваемое программное обеспечение, программы для микроконтроллеров

и процессоров с ограниченными ресурсами, вирусы, программные защиты и т.п.);

- быстродействие (программы, написанные на языке ассемблера выполняются гораздо быстрее, чем программы-аналоги, написанные на языках программирования высокого уровня абстракции. В данном случае быстродействие зависит от понимания того, как работает конкретная модель процессора, реальный конвейер на процессоре, размер кэша, тонкостей работы операционной системы. В результате, программа начинает работать быстрее, но теряет переносимость и универсальность).

Кроме того, знание языка ассемблера облегчает понимание архитектуры компьютера и работы его аппаратной части, то, чего не может дать знание языков высокого уровня абстракции (ЯВУ). В настоящее время большинство программистов разрабатывает программы в средах быстрого проектирования (Rapid Application Development) когда все необходимые элементы оформления и управления создаются с помощью готовых визуальных компонентов. Это существенно упрощает процесс программирования. Однако, нередко приходится сталкиваться с такими ситуациями, когда наиболее мощное и эффективное функционирование отдельных программных модулей возможно только в случае написания их на языке ассемблера (ассемблерные вставки). В частности, в любой программе, связанной с выполнением многократно повторяющихся циклических процедур, будь это циклы математических вычислений или вывод графических изображений, целесообразно наиболее времяемкие операции сгруппировать в программируемые на языке ассемблера субмодули. Это допускают все пакеты современных языков программирования высокого уровня абстракции, а результатом всегда является существенное повышение быстродействия программ.

Языки программирования высокого уровня абстракции разрабатывались с целью возможно большего приближения способа записи программ к привычным для пользователей компьютеров тех или иных форм

записи, в частности математических выражений, а также чтобы не учитывать в программах специфические технические особенности отдельных компьютеров. Язык ассемблера разрабатывается с учетом специфики процессора, поэтому для грамотного написания программы на языке ассемблера требуется, в общем, знать архитектуру процессора используемого компьютера.

Форт (от англ. forth — вперед, получившееся из-за необходимости сократить слово fourth — четвертый, до пяти букв из-за ограничений операционной системы на длину имени файла) в 1976 году стал стандартным языком для программирования для американского астрономического общества. Благодаря высокой степени переносимости и компактности, а также быстродействию, Форт до 1990х активно применялся во множестве приборов с микроконтроллерным управлением (спектрограф на Шаттле, микромодули управления искусственными спутниками Земли, для системы управления в аэропорту Эр-Рияда, системах компьютерного зрения, автоматизации анализа крови и кардиологического контроля, карманных переводчиках) [6]. Форт выступает прародителем такого широко известного языка управления печатью как PostScript.

По своей структуре Форт представляет собой набор примитивов — слов, которые участвуют в определении других слов. Важная особенность Форта — использование стека для передачи параметров между словами, такая конструкция позволяет очень гибко и просто реализовывать сложные концепции. Наборы слов, относящиеся к определенной области, могут выделяться в словари. Иерархическая структура словарей позволяет естественно организовывать наследование слов от словаря-родителя. Базовый словарь форта составляет менее четырех десятков слов, и уже он позволяет получать полноценные программы расширением исходного словаря.

Форт сам для себя является метасистемой — в нем объединены функции компилятора и интерпретатора, Форт может использоваться без

поддержки со стороны операционной системы и использоваться для того, чтобы компилировать самого себя, в том числе и на другие платформы.

В отличие от большинства языков программирования, которые имеют жесткую структуру, не позволяющую изменять синтаксис и многие базовые элементы языка, ничто не мешает на Форте написать модификацию системы, понимающую синтаксис той предметной области, для которой пишется программа. Ярким примером является написание Форт-ассемблера для конкретного процессора, выполняемое за один рабочий день квалифицированным фортером. Мало того, такой ассемблер поддерживает структурное программирование и прозрачную вставку в ассемблерный код слов из словаря самого Форта. Таким же образом реализуется поддержка объектно-ориентированного программирования.

До сих пор популярна тема создания Форт-процессоров, система команд которых является машинным представлением наиболее часто используемых слов-примитивов. Благодаря минимализму базовой Форт-системы это легко реализуемо, тем более что программируемые матрицы логических элементов сейчас достигли таких размеров, что Форт-процессор интегрируется на оставшиеся после программирования основной логики 10% вентилей как бонус. Это что-то из разряда такого: «А в углу нашей маленькой и уютной детской комнатки незаметно расположился аквариум с бегемотом».

Легкость написания Форт-систем привела к огромному разнообразию существующих реализаций — не писал свой Форт только ленивый фортер. Большое количество систем, достаточно плохо совместимых, поскольку имеется три стандарта Форта: Форт-79, Форт-83 и ANSI-Форт-94. Стандарты, хотя и описывают необходимый минимум слов для реализации, но дают огромный простор для самостоятельных изменений, что повсеместно и происходит. Существуют реализации Форта написанные на Ассемблере, Си, Питоне, Паскале, Яве, а также созданные с помощью целевых компиляторов других Форт-систем.

Как не хотелось бы радостно заявлять, что Форт живет полноценной жизнью языка программирования, однако это не совсем так. Форт прочно занял нишу как язык программирования микроконтроллеров, для которых надо организовать довольно сложную схему функционирования при минимуме ресурсов. Однако, примеры написания в современных условиях больших программ немногочисленны, поскольку скорость разработки на системах RAD превышает возможности Форт-систем.

Таким образом, по частоте использования, высокоуровневый язык программирования Форт находится даже ниже уровня Ассемблера. При решении проблем, требующих написания ассемблерного кода Форт имеет много преимуществ: компактность кода, структурный ассемблер, позволяющий легко внедрять в тело ассемблерных слов высокоуровневые определения. Для тех, кто хочет изучать и использовать Ассемблер, я бы порекомендовал начать это с изучения Форта, как я в свое время и сделал. Результатом такого метода обучения будет намного более быстрое и качественное освоение Ассемблера, тем более что после изучения основ вам обязательно захочется сделать свою Форт-систему (как ни говори, практика — кузница умения).

Язык С является наиболее известным и часто используемым в программировании с 70-х годов XX столетия. Структура данного языка похожа на структуру машинного и ассемблера. В связи с этим, Си активно применяют в процессе создания операционных систем, драйверов, системного программного обеспечения [2].

Нередко данный тип относят к языкам высокого уровня, однако, согласно определению, он относится к низкоуровневым языкам. Отнесение языка к той или иной группе определяется его назначением. Исходя из набора поддерживаемых команд, Си можно сопоставить с языками низкого уровня.

1.3. Преимущества и недостатки низкоуровневых языков.

Низкоуровневые языки обычно применяют для создания системных программ небольшого объема, драйверных устройств, стыковых модулей для нестандартного оборудования, в процессе программирования специальных микропроцессоров. Это оптимальное решение в тех случаях, когда в приоритете такие качества, как компактность, быстродействие, обеспечение прямого доступа к аппаратным ресурсам.

Преимущества языков низкого уровня:

- возможность создания эффективных программ;
- компактность;
- доступ ко всем возможностям процессора.

Недостатки низкоуровневых языков:

- высокие требования к квалификации программиста;
- необходимость в понимании устройства микропроцессорной системы, для которой предполагается написание программы;
- отсутствие возможности для переноса результирующей программы на компьютер или устройство, отличающееся типом процессора;
- на разработку больших и сложных программ необходимо затратить много времени.

Так как программы, созданные на языке низкого уровня, не требуют интерпретации или компиляции, они характеризуются большей скоростью по сравнению с аналогами, написанными на средне- и высокоуровневых языках. В этом случае программы взаимодействуют напрямую с регистрами и памятью. Низкоуровневые языки отличаются от других высокой эффективностью, что можно объяснить потреблением меньшего объема памяти.

С точки зрения работы с языком, низкоуровневые типы характеризуются повышенной сложностью. Программисты нередко испытывают трудности с бинарным кодом и мнемоникой. Языки низкого

уровня более технические по сравнению с другими, так как конкретная инструкция пишется под определенную архитектуру компьютера.

В связи с зависимостью от машин, низкоуровневые языки менее портируемые в отличие от средне- и высокоуровневых. Такое понятие, как абстракция, является отношением между языком и аппаратной частью компьютера. В случае с языками низкого уровня данный показатель минимален, либо отсутствует.

2. Выбор средств разработки.

2.1. Выбор языка программирования.

Для разработки данного проекта был выбран язык Си, так как он обладает значительным рядом преимуществ. Вот несколько из них:

- С сочетает в себе возможности языков высокого и низкого уровня. Он может использоваться для низкоуровневого программирования, такого как сценарии для драйверов и ядер, а также поддерживает функции языков программирования высокого уровня, таких как сценарии для программных приложений и т. д.
- С — это язык структурированного программирования, который позволяет разбить сложную программу на более простые программы, называемые функциями. Это также позволяет свободное перемещение данных через эти функции.
- Различные функции С, в том числе прямой доступ к аппаратным API-интерфейсам уровня машины, наличие компиляторов С, детерминированное использование ресурсов и динамическое распределение памяти, делают язык С оптимальным выбором для приложений сценариев и драйверов встроенных систем [3].
- С обладает высокой переносимостью и используется для создания сценариев системных приложений, которые составляют основную часть операционной системы Windows, UNIX и Linux.
- С — это язык программирования общего назначения, который может эффективно работать с корпоративными приложениями, играми, графикой и приложениями, требующими вычислений и т. д.
- Язык Си имеет богатую библиотеку, которая предоставляет ряд встроенных функций. Он также предлагает динамическое распределение памяти [1].
- С быстро реализует алгоритмы и структуры данных, ускоряя вычисления в программах.

2.2. Выбор среды разработки.

В качестве среды разработки для C приложения была выбрана CodeBlocks. Данная среда разработки обладает целым рядом очень полезных свойств, таких как:

- свободная лицензия GPL v3.0, в частности, разрешается бесплатное распространение и использование;
- среда может работать в операционных системах семейств Windows, Linux, OS X (то есть является кросс-платформенной);
- возможность работы с различными компиляторами;
- имеет компактное ядро и возможность расширения функционала посредством множества плагинов [4];
- Компактная и интуитивно понятная структура меню, обеспечивающая быструю настройку среды.

Именно эти преимущества CodeBlocks и помогли мне принять решение в выборе среды разработки.

3. Разработка игры викторины на языке С.

3.1. Анализ технического задания.

Требуется разработать игру викторину на языке С в среде разработки CodeBlocks. В рамках реализации проекта я поставила для себя несколько задач. Они представлены на рисунке 1.

Рисунок 1 – Блок-схема этапов выполнения задач для разработки

3.2. Результат разработки.

По окончании разработки была создана игра викторина, написанная на языке С. Меню игры включает в себя приветствие с игроком, а также информирует его о кнопках управления. Код, отвечающий за создание меню, находится на рисунке 2.

```
int main()
{
    int countr, r, r1, count, i, n;
    float score;
    char choice;
    char playername[20];
mainhome:
    system("cls");
    printf("\t\t\tC PROGRAM QUIZ GAME\n");
    printf("\n\t\t_____________________________________________");

    printf("\n\t\t\t   WELCOME ");
    printf("\n\t\t\t      to ");
    printf("\n\t\t\t   THE GAME ");
    printf("\n\t\t_____________________________________________");
    printf("\n\t\t_____________________________________________");
    printf("\n\t\t   BECOME A MILLIONAIRE!!!!!!!!!!    ");
    printf("\n\t\t_____________________________________________");
    printf("\n\t\t_____________________________________________");
    printf("\n\t\t > Press S to start the game");
    printf("\n\t\t > Press V to view the highest score  ");
    printf("\n\t\t > Press R to reset score");
    printf("\n\t\t > press H for help           ");
    printf("\n\t\t > press Q to quit            ");
    printf("\n\t\t_____________________________________________\n\n");
    choice = toupper(getch());
```

Рисунок 2 – Меню экрана на С

На следующем рисунке под номером 3 можно убедиться, что код работает.

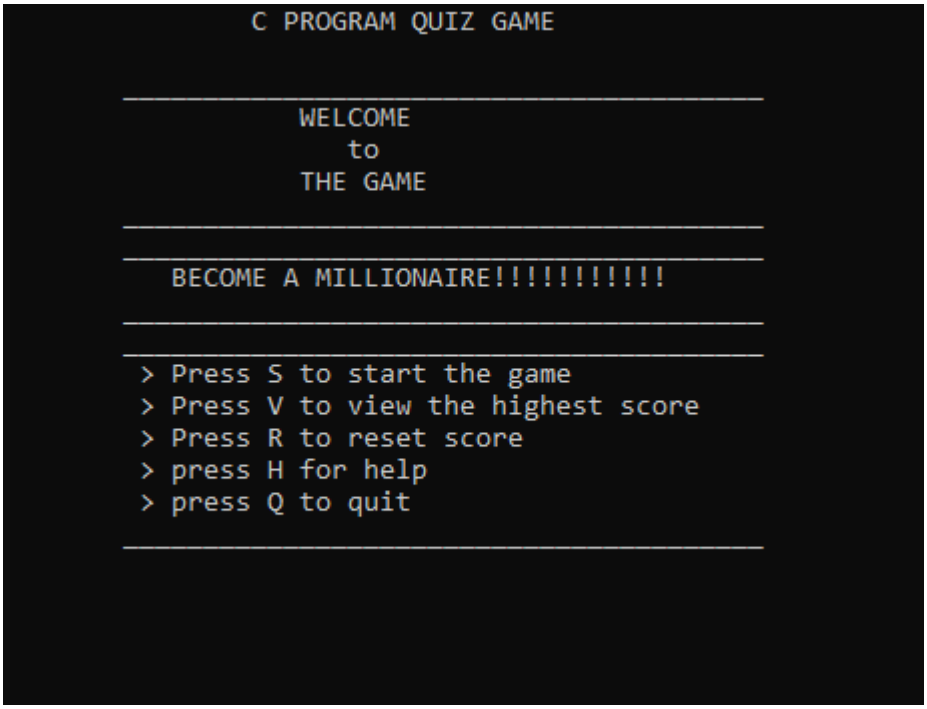

Рисунок 3 – Игровое меню

Если игроку понадобится правила игры, то он всегда сможет прочитать их при вводе “H”. Правила игры представлены на рисунке 4.

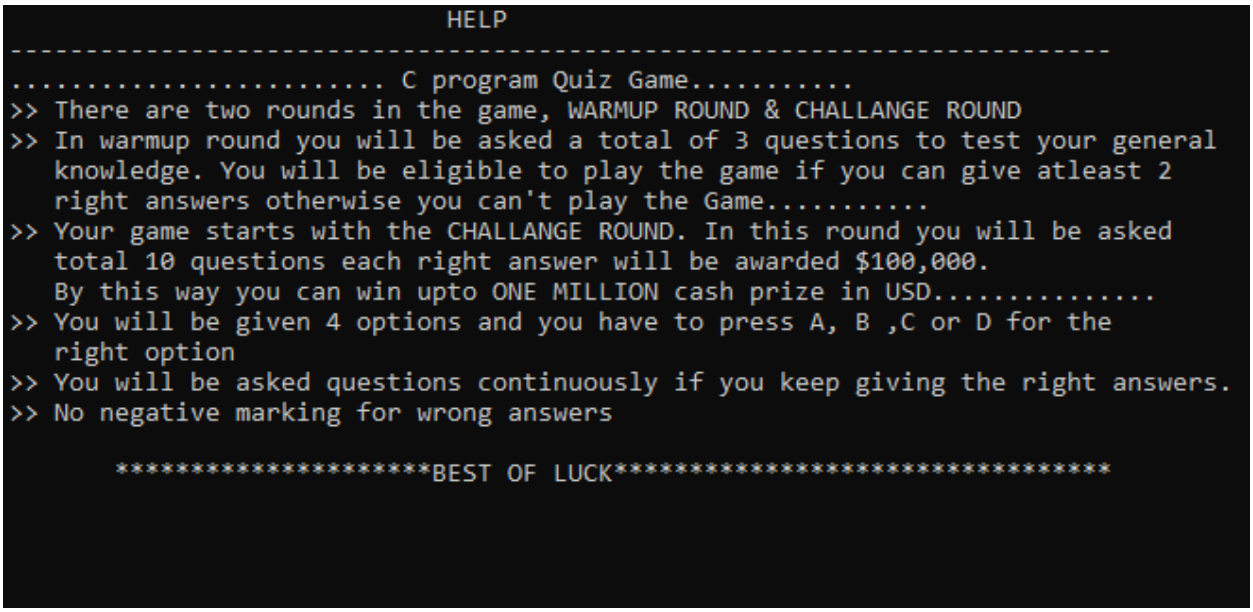

Рисунок 4 – Правила игры

После того, как игрок вводит “S”, на экран выводятся сами вопросы. Код, отвечающий за вывод вопросов расположен рисунке 5.

```
countr = 0;
for (i = 1; i <= 10; i++)
{
    system("cls");
    r = i;

    switch (r)
    {
    case 1:
        printf("\n\nWhat is the National Game of England?");
        printf("\n\nA.Football\t\tB.Basketball\n\nC.Cricket\t\tD.Baseball");
        if (toupper(getch()) == 'C')
        {
            printf("\n\nCorrect!!!");
            countr++;
            getch();
            break;
            getch();
        }
        else
        {
            printf("\n\nWrong!!! The correct answer is C.Cricket");
            getch();
            goto score;
            break;
        }

    case 2:
        printf("\n\n\nStudy of Earthquake is called.....,");
        printf("\n\nA.Seismology\t\tB.Cosmology\n\nC.Orology\t\tD.Etimology");
        if (toupper(getch()) == 'A')
        {
            printf("\n\nCorrect!!!");
            countr++;
            getch();
            break;
        }
        else
        {
            printf("\n\nWrong!!! The correct answer is A.Seismology");
            getch();
            goto score;
            break;
        }
```

Рисунок 5 – Вопросы на С

Как этот код работает можно увидеть на рисунке 6.

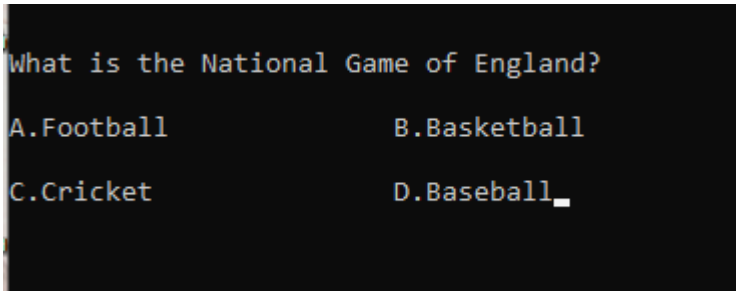

Рисунок 6 – Вопросы

В течении 6 первых вопросов, если игроком дано два правильных ответа или более игрок может перейти к следующему раунду, если менее двух, то игрок выбывает. Код этому условию представлен на рисунке 7.

```
        case 6:
            printf("\n\n\nWhat is the group of frogs known as?");
            printf("\n\nA.A traffic\t\tB.A toddler\n\nC.A police\t\tD.An Army");
            if (toupper(getch()) == 'D')
            {
                printf("\n\nCorrect!!!");
                count++;
                getch();
                break;
            }
            else
            {                                                    (char [43])"\n\nWrong!!! The correct answer is D.An A
                printf("\n\nWrong!!! The correct answer is D.An Army");
                getch();
                break;
            }
        }
    }

    if (count >= 2)
    {
        goto test;
    }
    else
    {
        system("cls");
        printf("\n\nSORRY YOU ARE NOT ELIGIBLE TO PLAY THIS GAME, BETTER LUCK NEXT TIME");
        getch();
        goto mainhome;
    }
test:
    system("cls");
    printf("\n\n\t*** CONGRATULATION %s you are eligible to play the Game ***", playername);
    printf("\n\n\n\n\t!Press any key to Start the Game!");
    if (toupper(getch()) == 'p')
    {
        goto game;
    }
```

Рисунок 7 – Тестовый раунд на С

Результат выполнения кода представлен на рисунке 8.

```
*** CONGRATULATION sds you are eligible to play the Game ***


!Press any key to Start the Game!
```

Рисунок 8 – Результат прохождения тестового раунда

Если же игрок продолжает игру после тестовых вопросов, то каждый последующий правильный ответ он будет получать по 100000. После 10 правильных вопросов подряд игрок становится миллионером и одерживает победу. Код для этой возможности можно увидеть на рисунке 9.

```
score:
    system("cls");
    score = (float)countr * 100000;
    if (score > 0.00 && score < 1000000)
    {
        printf("\n\n\t\t********** CONGRATULATION **********");
        printf("\n\t You won $%.2f", score);
        goto go;
    }

    else if (score == 1000000.00)
    {
        printf("\n\n\n \t\t********** CONGRATULATION **********");
        printf("\n\t\t\t\t YOU ARE A MILLIONAIRE!!!!!!!!!");
        printf("\n\t\t You won $%.2f", score);
        printf("\t\t Thank You!!");
    }
    else
    {
        printf("\n\n\t******** SORRY YOU DIDN'T WIN ANY CASH ********");
        printf("\n\t\t Thanks for your participation");
        printf("\n\t\t TRY AGAIN");
        goto go;
    }
```

Рисунок 9 – Выигрыш и проигрыш в игре на C

Результат работы этого куска кода можно увидеть на рисунке 10.

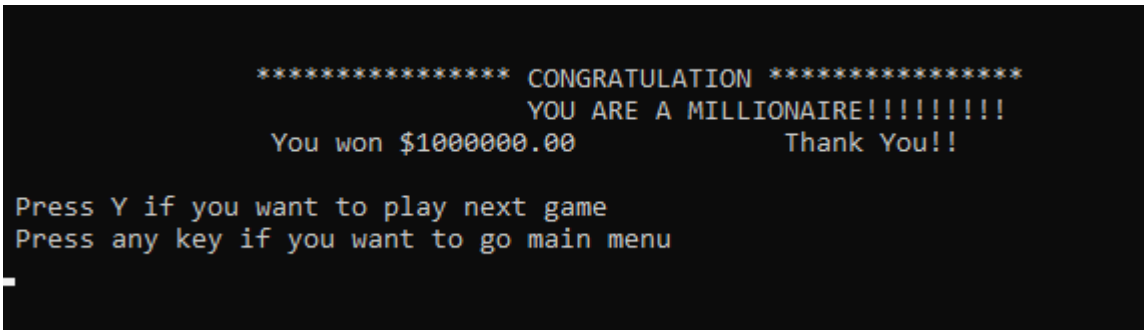

Рисунок 10 – Выигрыш в игре

В игре также присутствует функция сохранения результата конкретного игрока. Код для выполнения данной функции приложен на рисунке 11.

```
void show_record()
{
    system("cls");
    char name[20];
    float scr;
    FILE *f;
    f = fopen("score.txt", "r");
    fscanf(f, "%s%f", &name, &scr);
    printf("\n\n\t\t*************************************************************");
    printf("\n\n\t\t %s has secured the Highest Score %0.2f", name, scr);
    printf("\n\n\t\t*************************************************************");
    fclose(f);
    getch();
```

Рисунок 11 – Сохранение результата игрока на C

Сохранение происходит в обычный блокнот, однако, если кликнуть на “V” в меню, то можно увидеть результат, считанный из этого файла. Посмотреть результат можно на рисунке 12.

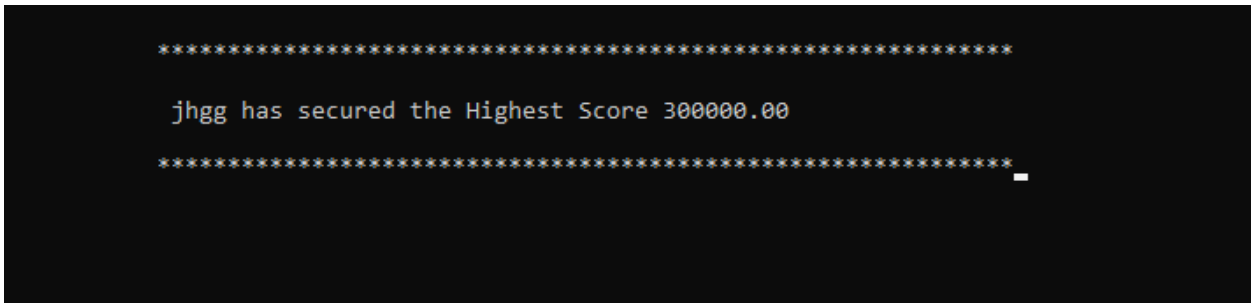

Рисунок 12 – Счет игрока

Заключение.

По итогу проделанной работы была разработана игра викторина на языке С.

В начале исследования было изучено понятие, характеристики и назначение низкоуровневого программирования, позже были изучены популярные языки, а также преимущества и недостатки языков данного вида.

Двигаясь дальше, появилась необходимость выбора средств разработки, а именно языка программирования и среды разработки продукта.

На последнем этапе предстояло проанализировать задачи, которые были поставлены для разработки и приступить к разработке.

На этапе разработки был создан удобный интерфейс игры, были созданы уровни игры, такие как “тестовый” и уровень основной игры. Была добавлена возможность набора очков на каждом этапе основной игры, а после набора 1000000 очков подряд игрок ставится победителем. Была также добавлена возможность сохранения счета конкретного игрока.

Все поставленные в введении задачи были выполнены.

Список использованных источников.

1. Васильев, А. Н. Программирование на С в примерах и задачах / А.Н. Васильев. – М.:ЭСКМО, 2017. – 560 с.

2. Подбельский В. В. Курс программирования на языке Си / В.В. Подбельский, С.С. Фомин. – М.:ДМК Пресс, 2018. – 384 с.

3. Гриффитс Д. Изучаем программирование на С / Д. Гриффитс. – М.:ЭСКМО, 2019. – 624 с.

4. CodeBlocks :: не просто ещё одна IDE – URL: http://we.easyelectronics.ru/CADSoft/codeblocks-ne-prosto-esche-odna-ide.html (дата обращения: 28.11.2021). – Текст: электронный.

5. Как писать на ассемблере в 2018 году – URL: https://habr.com/ru/post/345748/ (дата обращения: 28.11.2021). – Текст: электронный.

6. Язык программирования Форт – URL: https://habr.com/ru/post/29967/ (дата обращения: 28.11.2021). – Текст: электронный.