МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт цифровых технологий, электроники и физики (ИЦТЭФ)
Кафедра вычислительной техники и электроники (ВТиЭ)

Генератор персонажа Dangeons & Dragons на платформе Arduino

Отчет по научно-исследовательской работе

Выполнил: студент 595 гр.
_____ А. В. Лаптев
Проверил: ст. преп. кафедры ВТиЭ
_____ И. А. Шмаков
«____» _____ 2022 г.

Барнаул, 2022 г.

РЕФЕРАТ

Полный объём работы составляет 34 страницы, включая 11 рисунков, 0 таблиц и 1 листинг кода.

Цель работы — разработка программного продукта под микроконтроллер одного из рассмотренных семейств AVR или ARM, на выбранной программной и аппаратной платформах.

В результате выполнения научно-исследовательской работы было создано приложение для генерации персонажа Dangeous & Dragons с использованием аппаратно-программной платформы Arduino.

Ключевые слова: AVR, ARM, Arduino, микроконтроллер, генератор персонажа DnD.

Отчёт оформлена с помощью системы компьютерной вёрстки TEX и его расширения XETEX из дистрибутива *TeX Live*.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день практически у каждого есть какое-либо портативное устройство или другая потребительская электроника. В такой технике приоритетом является низкое энергопотребление в сочетании с относительным быстродействием. Этим требованиям отвечают семейства микроконтроллеров AVR и ARM.

Сейчас встраиваемые системы и портативные устройства — одна из самых быстрорастущих технологических сфер. И, поэтому, разработка под микроконтроллеры и микропроцессоры этих семейств является актуальной.

Целью данной работы является разработка программного продукта под микроконтроллер одного из рассмотренных семейств, с использованием выбранной программной и аппаратной платформ.

Задачи работы:

1. Рассмотрение имеющихся аппаратных и программных средств для выбора наиболее подходящей платформы для разработки под AVR и ARM микроконтроллеры.
2. Разработка собственного программного продукта на выбранной аппаратно-программной платформе.

1. КРАТКИЙ ОБЗОР ПЛАТФОРМ

Для облегчения разработки под микроконтроллеры имеет смысл использовать готовые отладочные платы, поскольку на них есть не только микроконтроллер, но и набор обвязки, необходимый для его нормальной работы.

На сегодняшний день существует довольно много аппаратных и программных платформ для разработки под микроконтроллеры.

Среди представителей в аппаратной части можно выделить следующих:

1. LaunchPad — аппаратная платформа, включающая в себя платы с микроконтроллером и платы расширения, производства Texas Instruments. Платформа позиционируется как конкурент Arduino. Специально под LaunchPad была разработана программная платформа Enrgia [1; 2].
2. STM Nucleo — семейство отладочных плат от STMicroelectronics на основе микроконтроллеров STM32. Функциональность плат может быть расширена путем подключения плат расширения для Arduino, поскольку она совместима со многими ее компонентами [3; 4].
3. Mbed — аппаратная часть имеет в своем составе отладочные платы от ARM и других компаний: семейства плат mbed и FRDM от NXP Semiconductors, семейство Nucleo от STMicroelectronics, семейство EFM32 от Silicon Labs и другие [5].
4. Arduino — аппаратная часть представляет собой набор плат с микроконтроллером и платы расширения. Большинство плат с микроконтроллером снабжено минимально необходимым набором обвязки для нормальной работы микроконтроллера. Сторонними производителями выпускаются различные датчики и устройства, совместимые с Arduino [6].

Среди представителей в программной части существуют следующие варианты:

1. Atmel Studio — интегрированная платформа разработки, предназначенная для проектирования и отладки приложений для микроконтроллеров Atmel на базе микроконтроллеров AVR и ARM. В Atmel Studio поддерживаются следующие языки программирования: C/C++, ассемблер [7]. IDE имеет все необходимое для комфортного и быстрого написания кода, а также, является гибко настраиваемой. Данное решение является абсолютно бесплатным [8].
2. MPLAB — интегрированная среда разработки для контроллеров производства Microchip. Данная среда поддерживает разработку программ,

написанных на C и ассемблере. Имеет множество дополнительных функций и модулей, которые упрощают процесс разработки, отладки и тестирования программ и папку с шаблонами программ на ассемблере, с которыми удобно начинать работу. Распространяется бесплатно [9].

3. Mbed (онлайн IDE) — в программной части, платформа включает в себя интегрированную среду разработки, которая работает онлайн. Среда включает в себя компилятор, набор библиотек, текстовый редактор и примеры программного кода. Поддерживается облачное хранение кода системой контроля версий Mercurial [5].
4. Arduino IDE — программная часть платформы, которая представляет собой бесплатную IDE, в которую помимо инструментов для разработки включены также инструменты для загрузки программы в микроконтроллер. В основе среды лежит фреймворк Wiring, оболочка написана на основе проекта Processing на Java [6].
5. Energia — программная платформа для прототипирования электроники с открытым исходным кодом, создана для интеграции с аппаратной платформой LaunchPad от Texas Instruments. Как и Arduino IDE, использует в своей основе фреймворк Wiring и оболочку, написанную на основе проекта Processing, поэтому их интерфейсы схожи [10; 11].

Среди всех этих представителей выделяется платформа Arduino, которая, в свою очередь, включает в себя как программную платформу для разработки под микроконтроллеры (среда разработки Arduino IDE), так и аппаратную (большой выбор отладочных плат с различными микроконтроллерами в своем составе,а также плат расширения к ним).

Помимо этого, для программирования в данной платформе используется модифицированный язык C, который называют Arduino C, который довольно хорошо описан на сегодняшний день и проще в освоении, чем чистый C.

Также, платформа имеет довольно большое сообщество разработчиков и производителей аппаратной части, благодаря которому существует множество сторонних библиотек и Arduino-совместимых плат от других производителей, что делает разработку на этой платформе более доступной и комфортной.

Благодаря этому, данная платформа больше подходит для людей, которые не имеют достаточного опыта разработки под микроконтроллеры, в связи с чем эта платформа и была выбрана для реализации данного программного продукта.

2. ГЕНЕРАТОР ПЕРСОНАЖА DANGEONS & DRAGONS (DND)

2.1. Правила генерации

Генерация персонажа представляет собой совокупность из параметров персонажа, которые пользователь может выбрать из предложенных вариантов, а также характеристик, которые одинаковы для всех персонажей, но значения которых генерируются случайным образом.

Изначально, пользователю предоставляется возможность выбрать то, кем он, непосредственно, хочет быть в игре (раса и класс персонажа) [12]. После чего происходит генерация значений для характеристик персонажа.

Для определения значений характеристик персонажа существует несколько различных способов [13; 14]. В данном случае, характеристики персонажа генерируются следующим образом:

1. для каждой из характеристик производится четыре броска игральной кости;
2. меньшее из выброшенных значений исключается из генерации;
3. оставшиеся три значения суммируются. Полученное значение и будет являться значением для характеристики.

Для некоторых других характеристик персонажа, таких как хитпоинты и класс защиты, генерация осуществляется иначе.

Для расчета хит-поинтов для героя первого уровня берется максимальное значение кости хитов, которое определяется классом выбранного персонажа и модификатор телосложения, который рассчитывается исходя из соответствующей характеристики. Полученные значения складываются, что и дает количество хит-поинтов персонажа.

Для расчета класса защиты для героя первого уровня используется значение 10 (базовое значение для персонажа, который не носит броню) и модификатор ловкости, рассчитанный исходя из характеристики ловкости. Данные значения складываются, выводя значение, которое и определяет класс защиты персонажа.

Модификаторы телосложения, ловкости и других характеристик рассчитываются одинаково. Формула для расчета модификатора конкретной характеристики выглядит следующим образом: $MC = (CH - 10)/2$, где МС — модификатор характеристики, СН — конкретное значение характеристики, для которого производится расчет. Полученный результат округляется в меньшую сторону. Модификатор характеристики может принимать значение от -5 до $+10$ [15].

2.2. Реализация генератора персонажа DnD

Генератор персонажа реализован на отладочной плате Arduino Uno, в основе которой лежит микроконтроллер ATmega328. Для отображения меню генерации персонажа и навигации по нему используется плата расширения LCD Keypad Shield (Arduino-совместимая), на которой располагается LCD-дисплей 16x2, а также набор кнопок, которые могут быть использованы, непосредственно, для навигации. Разработка генератора персонажа велась с использованием интегрированной среды разработки Arduino IDE.

В генераторе реализованы следующие возможности:

1. возможность выбора расы персонажа.
2. Возможность выбора класса персонажа.
3. Генерация случайных значений для характеристик персонажа, согласно правилам DnD.

   Базовые характеристики персонажа DnD:
   - Сила (Strong — сокращенно «Str»);
   - Телосложение (Constitution — сокращенно «Con»);
   - Ловкость (Dexterity — сокращенно «Dex»);
   - Интеллект (Intelligence — сокращенно «Int»);
   - Мудрость (Wisdom — сокращенно «Wis»);
   - Харизма (Charisma — сокращенно «Cha»).
4. Расчет количества Хит-поинтов (Hit Points — сокращенно «HP») персонажа на основе сгенерированных характеристик.
5. Расчет Класса Защиты (Armor Class — сокращенно «AC») персонажа с использованием сгенерированных характеристик.

Для реализации данного набора возможностей решение было разбито на функции, каждая из которых отвечает за выбор параметров персонажа или за генерацию значений его характеристик.

2.2.1. Функция mainText

Эта функция представляет собой реализацию приветственного окна, в котором сообщается о назначении программы (генератор персонажа DnD) и указывается, что для продолжения работы нужно нажать соответствующую кнопку.

Помимо этого, после нажатия кнопки внутри функции генерируются псевдослучайные числа, которые имитируют броски игральной кости. Генерируется четыре псевдослучайных числа и находится сумма этих чисел, ис-

ключая меньшее. Такие манипуляции производятся шесть раз (по количеству характеристик персонажа). Блок-схема подпрограммы mainText изображена на рис. 2.4, 2.5 в Приложении А.

2.2.2. Функция genRacePers

В этой функции реализована часть генератора, которая отвечает за выбор пользователем расы персонажа. На экран выводятся двенадцать вариантов рас, среди которых пользователь может переключаться нажатием на кнопки: «ВВЕРХ» («UP»), «ВНИЗ» («DOWN»), «ВПРАВО» («RIGHT»), «ВЛЕВО» («LEFT»); чтобы подтвердить свой выбор требуется нажать кнопку «ВЫБРАТЬ» («SELECT»). Блок-схема подпрограммы genRacePers (см. рис. 2.6, 2.7 в Приложении А).

2.2.3. Функция genClassPers

В этой функции реализована возможность выбора класса персонажа. На экран последовательно выводятся тринадцать предусмотренных классов, между которыми также можно переключаться с помощью кнопок, для подтверждения выбора предусмотрена кнопка «ВЫБРАТЬ» («SELECT»). Блок-схема подпрограммы genClassPers изображена на рис. 2.8, 2.9 в Приложении А.

2.2.4. Функция charactPers

С помощью данной функции на экран выводятся значения характеристик, сгенерированные в функции mainText, а также на их основе высчитываются количество хит-поинтов и класс защиты для выбранных расы и класса, которые также выводятся на экран. Навигация здесь, также как и в предыдущих функциях осуществляется с помощью кнопок. Блок-схема подпрограммы charactPers изображена на рис. 2.10, 2.11 в Приложении А.

2.2.5. Функция clickButton

Эта функция необходима для того, чтобы обрабатывать нажатия кнопок, она возвращает целое число, которое будет указывать на то, нажата ли кнопка и, если нажата, то какая. Блок-схема подпрограммы clickButton изображена на рис. 2.3 в Приложении А.

Помимо этих функций в проектах, написанных в Arduino IDE есть еще две стандартных функции. В функции setup задается скорость передачи данных для последовательного порта и инициализируется lcd-дисплей. В функции loop вызывается функция mainText для того, чтобы генератор персонажа работал вплоть до отключения его от питания. Блок-схемы алгоритмов подпрограмм loop и setup изображены на рис. 2.1, 2.2 в Приложении А.

2.2.6. Недостатки разработанного генератора персонажа DnD

Полученный скетч использует 6395 байт памяти устройства, что составляет примерно 19 % от общего объема памяти. Глобальные переменные, в свою очередь, используют 678 байт динамической памяти, что составляет примерно 33 % от ее объема. Такие показатели не являются оптимальными для подобной программы и на то есть несколько причин.

Во-первых встроенные функции в Arduino IDE зачастую имеют в своем составе ряд дополнительных проверок, чтобы минимизировать количество возможных ошибок, которые могут появиться при компиляции, что, в свою очередь, расходует довольно значительную часть ресурсов [16].

Во-вторых, довольно большое количество глобальных переменных, а также их не оптимальная типизация, из-за которой они могут занимать в памяти больше места, чем им требуется в действительности, также увеличивают количество потребляемых ресурсов микроконтроллера и, как следствие, уменьшают общее быстродействие и производительность.

Также на работу программы негативное влияние оказывает использование встроенной функции задержки delay, поскольку, при использовании этой функции, приостанавливается работа всей программы [17]. Для задержек больше подходят прерывания либо другая функция — millis, которая указывает сколько времени нужно «обходить» тот блок кода, выполнение которого необходимо приостановить.

ЗАКЛЮЧЕНИЕ

В ходе выполнения научно-исследовательской работы было проведено знакомство с различными платформами для разработки под микроконтроллеры, проведен обзор различных семейств микроконтроллеров и выбрана платформа для разработки собственного программного продукта — генератора персонажа DnD.

Были решены основные поставленные задачи, среди которых:

1. наличие возможности выбора расы персонажа;
2. наличие возможности выбора класса персонажа;
3. генерация случайных значений для характеристик персонажа, согласно правилам DnD;
4. расчет некоторых дополнительных характеристик персонажа с использованием значений, сгенерированных для основных характеристик.

В результате чего, был разработан вариант генератора персонажа DnD.

Поскольку в данном генераторе реализованы лишь базовые возможности, то в будущем есть возможность для добавления в генератор ряда функций, например, выбор снаряжения на основе выбранного класса или расчет бросков атаки и урона персонажа, чтобы сделать генератор пригодным для полноценного создания персонажа и облегчения расчетов во время игры.

Кроме того, отладочная плата имеет довольно большие габариты и часть периферии не используется, в будущем планируется развести и вытравить собственную плату для уменьшения габаритов устройства, также возможна замена дисплея на больший по размерам, поскольку количество строк и символов в строке дисплея на плате расширения не позволяет пользователю комфортно работать с ним.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. [Электронный ресурс] Недорогая альтернатива Arduino — habr.com. — URL: https://habr.com/ru/post/151196.
2. [Электронный ресурс] Аппаратные комплекты и платы — ti.com. — URL: https://www.ti.com/design-resources/embedded-development/hardware-kits-boards.html.
3. [Электронный ресурс] Платы STM32 Nucleo — st.com. — URL: https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html.
4. [Электронный ресурс] Семейство STM32 NUCLEO — terraelectronica.ru. — URL: https://barnaul.terraelectronica.ru/news/4147#:~:text=STM32%20NUCLEO%20%2D%20%D0%BD%D0%BE%D0%B2%D0%B5%D0%B9%D1%88%D0%B5%D0%B5%20%D1%81%D0%B5%D0%BC%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%BE%20%D0%B1%D1%8E%D0%B4%D0%B6%D0%B5%D1%82%D0%BD%D1%8B%D1%85,%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D1%8B%20%D0%BD%D0%B0%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B5%20%D0%BC%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80%D0%BE%D0%B2%20STM32..
5. [Электронный ресурс] Mbed — Википедия. — URL: https://ru.wikipedia.org/wiki/Mbed#:~:text=Mbed%20%E2%80%94%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%2D%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F%20%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0%20%D0%B8,ARM%20%D1%81%D0%BE%D0%B2%D0%BC%D0%B5%D1%81%D1%82%D0%BD%D0%BE%20%D1%81%20%D0%B4%D1%80%D1%83%D0%B3%D0%B8%D0%BC%D0%B8%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8..
6. [Электронный ресурс] Arduino — Википедия. — URL: https://ru.wikipedia.org/wiki/Arduino#%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80.

7. [Электронный ресурс] Программы для микроконтроллеров — cxem.net. — URL: https://cxem.net/software/soft_mcu.php.
8. [Электронный ресурс] Atmel Studio — sites.google.com. — URL: https://sites.google.com/site/100voltsamper/sredy-razrabotok/atmel-studio.
9. [Электронный ресурс] MPLAB — Википедия. — URL: https://ru.wikipedia.org/wiki/MPLAB.
10. [Электронный ресурс] Energia — energia.nu. — URL: https://energia.nu.
11. [Электронный ресурс] Energia IDE — ti.com. — URL: https://www.ti.com/tool/ENERGIA.
12. [Электронный ресурс] Классы D&D 5 - dnd.su. — URL: https://dnd.su/class.
13. [Электронный ресурс] Создание персонажа для Dangeons and Dragons — longstoryshort.app. — URL: https://longstoryshort.app/long/character-creation.
14. [Электронный ресурс] Создание персонажа - dungeon.fandom.com. — URL: https://dungeon.fandom.com/ru/wiki/%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%B6%D0%B0#.D0.A0.D0.B0.D1.81.D0.BF.D1.80.D0.B5.D0.B4.D0.B5.D0.BB.D0.B5.D0.BD.D0.B8.D0.B5_.D1.87.D0.B8.D1.81.D0.B5.D0.BB.
15. [Электронный ресурс] Основные формулы - dnd.su. — URL: https://dnd.su/articles/26-osnovnye_formuly.
16. [Электронный ресурс] Почему мнгоие не любят Arduino — habr.com. — URL: https://habr.com/ru/post/254163.
17. [Электронный ресурс] Arduino delay millis и micros для организации задержки в скетче — arduinomaster.ru. — URL: https://arduinomaster.ru/program/arduino-delay-millis/#:~:text=%D0%9D%D1%83%D0%B6%D0%BD%D0%BE%20%D0%BE%D1%82%D1%87%D0%B5%D1%82%D0%BB%D0%B8%D0%B2%D0%BE%20%D0%BF%D0%BE%D0%BD%D0%B8%D0%BC%D0%B0%D1%82%D1%8C%2C%20%D1%87%D1%82%D0%BE%20%D0%BD%D0%B0,%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8%20delay%20%D0%B2%20Arduino..

ПРИЛОЖЕНИЕ

Листинг 2.1 Исходный код приложения

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // Пины для подключения lcd-дисплея

#define BTN_R 1
#define BTN_U 2
#define BTN_D 3
#define BTN_L 4
#define BTN_S 5
#define BTN_NONE 10

int i = 0;
int j = 0;
int k = 0;
int sel;
int characteristics[] = {0, 0, 0, 0, 0, 0, 0, 0};
char* characteristicsPers[] = {"Race: ", "Class: ", "Str: ", "Con: ", "Dex: ", "Int: ", "Wis: ", "Cha: ", "HP: ", "AC: "};
char* racePers[] = {"Human", "Dwarf", "Elf", "Half-orc", "Gnome", "Goliath", "Halfling", "Genasi", "Tiefling", "Aasimar", "Eladrin", "Dragonborn"};
char* classPers[] = {"Bard", "Barbarian", "Fighter", "Wizard", "Druid", "Cleric", "Artificer", "Warlock", "Monk", "Paladin", "Rogue", "Ranger", "Sorcerer"};
char* pers[] = {"", ""};

//Стрелка вврех
byte arrowUp[8] = {
  0b00000,
```

```
25    0b00000,
26    0b00100,
27    0b01110,
28    0b11111,
29    0b00000,
30    0b00000,
31    0b00000
32  };
33
34  //Стрелка вниз
35  byte arrowDown[8] = {
36    0b00000,
37    0b00000,
38    0b00000,
39    0b11111,
40    0b01110,
41    0b00100,
42    0b00000,
43    0b00000
44  };
45
46  //Обработка нажатой клавиши
47  int clickButton() {
48    int keyAnalog = analogRead(A0);
49
50    if (keyAnalog < 100) {
51      // Значение меньше 100 - нажата кнопка right
52      return BTN_R;
53    } else if (keyAnalog < 200) {
54      // Значение больше 100 (иначе мы бы вошли в
        ↪  предыдущий блок результата сравнения, но
        ↪  меньше 200 - нажата кнопка UP
55      return BTN_U;
56    } else if (keyAnalog < 400) {
57      // Значение больше 200, но меньше 400 - нажата
        ↪  кнопка DOWN
58      return BTN_D;
```

```
59    } else if (keyAnalog < 600) {
60      // Значение больше 400, но меньше 600 - нажата
        ↪  кнопка LEFT
61      return BTN_L;
62    } else if (keyAnalog < 800) {
63      // Значение больше 600, но меньше 800 - нажата
        ↪  кнопка SELECT
64      return BTN_S;
65    } else {
66      // Все остальные значения (до 1023) будут
        ↪  означать, что нажатий не было
67      return BTN_NONE;
68    }
69  }
70
71  //Текст в главном окне
72  void mainText() {
73    int roll;
74    int randRoll = 0;
75    int characteristic;
76
77    lcd.setCursor(0, 1);
78    lcd.print("Press the 'SELECT' button");
79    lcd.home();
80    lcd.print("Character generator DnD");
81    delay(500);
82
83    int btnSel = 0;
84    for (int j = 0; j < 9; j++) {
85      btnSel = clickButton();
86      if (btnSel == 5) {
87        break;
88      }
89      lcd.scrollDisplayLeft();
90      delay(500);
91    }
92
```

```
93    switch (btnSel){  //Действия при нажатии кнопки
       ↪  SELECT
94      case BTN_S:
95        randomSeed(millis()); // Генерируем
           ↪  псевдослучайное число (каждый раз различное)
96        for (int i = 0; i < 6; i++) { //Определяем
           ↪  значение для характеристики
97          roll = 6;
98          characteristic = 0;
99          for (int j = 0; j < 4; j++) { //Для определения
             ↪  одной характеристики надо провести 6
             ↪  бросков
100           randRoll = random(1, 7);  //Симулируем бросок
               ↪  игральной кости
101           if (randRoll < roll) {  //Осуществляем поиск
               ↪  минимального значения среди выброшенных
102             roll = randRoll;
103             characteristic += randRoll;
104           }
105           else
106           {
107             characteristic += randRoll;
108           }
109         }
110         characteristics[i] = characteristic - roll;
             ↪  //Отбрасываем наименьшее значение
111       }
112       delay(500);
113       lcd.clear();
114       while(true) {
115         lcd.home();
116         genRacePers();  //Вызов функции для выбора расы
117       }
118       break;
119     default:
120       break;
121   }
```

```
122   lcd.clear();
123 }
124
125 //Выбор расы персонажа
126 void genRacePers() {
127   if(i == 0) {
128     lcd.print("Select race:");
129     lcd.setCursor(0, 1);
130     lcd.print(racePers[i]);
131     lcd.createChar(2, arrowDown);
132     lcd.setCursor(15, 1);
133     lcd.print(char(2));
134   }
135   else if(i == 11) {
136     lcd.print("Select race:");
137     lcd.setCursor(0, 1);
138     lcd.print(racePers[i]);
139     lcd.createChar(1, arrowUp);
140     lcd.setCursor(15, 0);
141     lcd.print(char(1));
142   }
143   else {
144     lcd.print("Select race:");
145     lcd.setCursor(0, 1);
146     lcd.print(racePers[i]);
147     lcd.createChar(1, arrowUp);
148     lcd.setCursor(15, 0);
149     lcd.print(char(1));
150     lcd.createChar(2, arrowDown);
151     lcd.setCursor(15, 1);
152     lcd.print(char(2));
153   }
154   delay(150);
155
156   int btnRacePers = clickButton();
157   switch (btnRacePers){
158     case BTN_R:
```

```
159         lcd.clear();
160         i++;
161         if (i > 11)
162           i--;
163         break;
164       case BTN_U:
165         lcd.clear();
166         i--;
167         if (i < 0)
168           i++;
169         break;
170       case BTN_D:
171         lcd.clear();
172         i++;
173         if (i > 11)
174           i--;
175         break;
176       case BTN_L:
177         lcd.clear();
178         i--;
179         if (i < 0)
180           i++;
181         break;
182       case BTN_S:
183         pers[0] = racePers[i];  //Запоминаем выбранную
              ↪  расу
184         lcd.clear();
185         while(true) {
186           lcd.home();
187           genClassPers(); //Вызываем функцию выбора
                ↪  класса
188         }
189         break;
190       default:
191         break;
192     }
193     lcd.clear();
```

```
194 }
195
196 //Функция выбора класса персонажа
197 void genClassPers() {
198   if(j == 0) {
199     lcd.print("Select class:");
200     lcd.setCursor(0, 1);
201     lcd.print(classPers[j]);
202     lcd.createChar(2, arrowDown);
203     lcd.setCursor(15, 1);
204     lcd.print(char(2));
205   }
206   else if(j == 12) {
207     lcd.print("Select class:");
208     lcd.setCursor(0, 1);
209     lcd.print(classPers[j]);
210     lcd.createChar(1, arrowUp);
211     lcd.setCursor(15, 0);
212     lcd.print(char(1));
213   }
214   else {
215     lcd.print("Select class:");
216     lcd.setCursor(0, 1);
217     lcd.print(classPers[j]);
218     lcd.createChar(1, arrowUp);
219     lcd.setCursor(15, 0);
220     lcd.print(char(1));
221     lcd.createChar(2, arrowDown);
222     lcd.setCursor(15, 1);
223     lcd.print(char(2));
224   }
225   delay(150);
226
227   int btnClassPers = clickButton();
228   switch (btnClassPers){
229     case BTN_R:
230       lcd.clear();
```

```
231         j++;
232         if (j > 12)
233           j--;
234         break;
235       case BTN_U:
236         lcd.clear();
237         j--;
238         if (j < 0)
239           j++;
240         break;
241       case BTN_D:
242         lcd.clear();
243         j++;
244         if (j > 12)
245           j--;
246         break;
247       case BTN_L:
248         lcd.clear();
249         j--;
250         if (j < 0)
251           j++;
252         break;
253       case BTN_S:
254         pers[1] = classPers[j]; //Запоминаем выбранный
              ↪  класс
255         lcd.clear();
256         while(true) {
257           lcd.home();
258           charactPers();  //Вызываем функцию для
                ↪  просмотра сгенерированных характеристик
259         }
260         break;
261       default:
262         break;
263     }
264     lcd.clear();
265 }
```

```
266
267 //Функция для просмотра сгенерированных характеристик
268 void charactPers() {
269   int HP;
270   int AC;
271
272   //Определяем количество хит-поинтов персонажа
273   if ((pers[1] == classPers[0]) || (pers[1] ==
      ↪  classPers[4]) || (pers[1] == classPers[5]) ||
      ↪  (pers[1] == classPers[6]) || (pers[1] ==
      ↪  classPers[7]) || (pers[1] == classPers[8]) ||
      ↪  (pers[1] == classPers[10])) {
274     HP = 8 + floor((characteristics[1] - 10) / 2);
275   }
276   else if (pers[1] == classPers[1]) {
277     HP = 12 + floor((characteristics[1] - 10) / 2);
278   }
279   else if ((pers[1] == classPers[2]) || (pers[1] ==
      ↪  classPers[9]) || (pers[1] == classPers[11])) {
280     HP = 10 + floor((characteristics[1] - 10) / 2);
281   }
282   else {
283     HP = 6 + floor((characteristics[1] - 10) / 2);
284   }
285   characteristics[6] = HP;
286
287   //Определяем класс защиты персонажа
288   AC = 10 + floor((characteristics[2] - 10) / 2);
289   characteristics[7] = AC;
290
291   if(k == 0) {
292     lcd.print("Characteristics:");
293     lcd.setCursor(0, 1);
294     lcd.print(characteristicsPers[k]);
295     lcd.print(pers[k]);
296     lcd.createChar(2, arrowDown);
297     lcd.setCursor(15, 1);
```

```
298       lcd.print(char(2));
299     }
300     else if(k == 1) {
301       lcd.print("Characteristics:");
302       lcd.setCursor(0, 1);
303       lcd.print(characteristicsPers[k]);
304       lcd.print(pers[k]);
305       lcd.createChar(1, arrowUp);
306       lcd.setCursor(15, 0);
307       lcd.print(char(1));
308       lcd.createChar(2, arrowDown);
309       lcd.setCursor(15, 1);
310       lcd.print(char(2));
311     }
312     else if(k == 9) {
313       lcd.print("Characteristics:");
314       lcd.setCursor(0, 1);
315       lcd.print(characteristicsPers[k]);
316       lcd.print(characteristics[k - 2]);
317       lcd.createChar(1, arrowUp);
318       lcd.setCursor(15, 0);
319       lcd.print(char(1));
320     }
321     else {
322       lcd.print("Characteristics:");
323       lcd.setCursor(0, 1);
324       lcd.print(characteristicsPers[k]);
325       lcd.print(characteristics[k - 2]);
326       lcd.createChar(1, arrowUp);
327       lcd.setCursor(15, 0);
328       lcd.print(char(1));
329       lcd.createChar(2, arrowDown);
330       lcd.setCursor(15, 1);
331       lcd.print(char(2));
332     }
333     delay(150);
334
```

```
335   int btnCharPers = clickButton();
336   switch (btnCharPers){
337     case BTN_R:
338       lcd.clear();
339       k++;
340       if (k > 9)
341         k--;
342       break;
343     case BTN_U:
344       lcd.clear();
345       k--;
346       if (k < 0)
347         k++;
348       break;
349     case BTN_D:
350       lcd.clear();
351       k++;
352       if (k > 9)
353         k--;
354       break;
355     case BTN_L:
356       lcd.clear();
357       k--;
358       if (k < 0)
359         k++;
360       break;
361     default:
362       break;
363   }
364   lcd.clear();
365 }
366
367 void setup() {
368   lcd.begin(16, 2); // Инициализация текстового
      ↪  дисплея 16x2
369 }
370
```

```
371 void loop() {
372   mainText(); //Вызов функции для отображения текста
      ↪  на главном экране
373 }
```

Блок-схемы подпрограмм

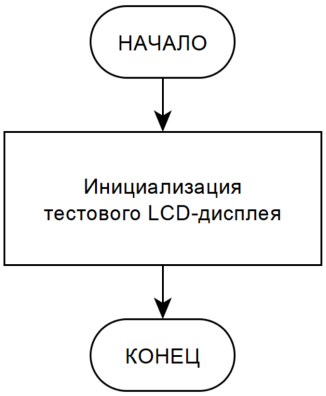

Рис. 2.1 Блок-схема функции setup

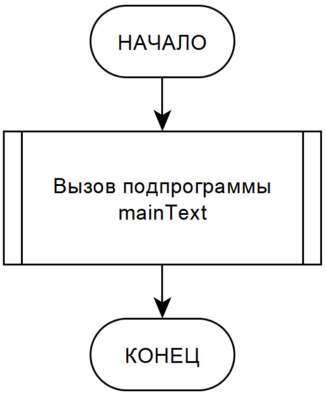

Рис. 2.2 Блок-схема функции loop

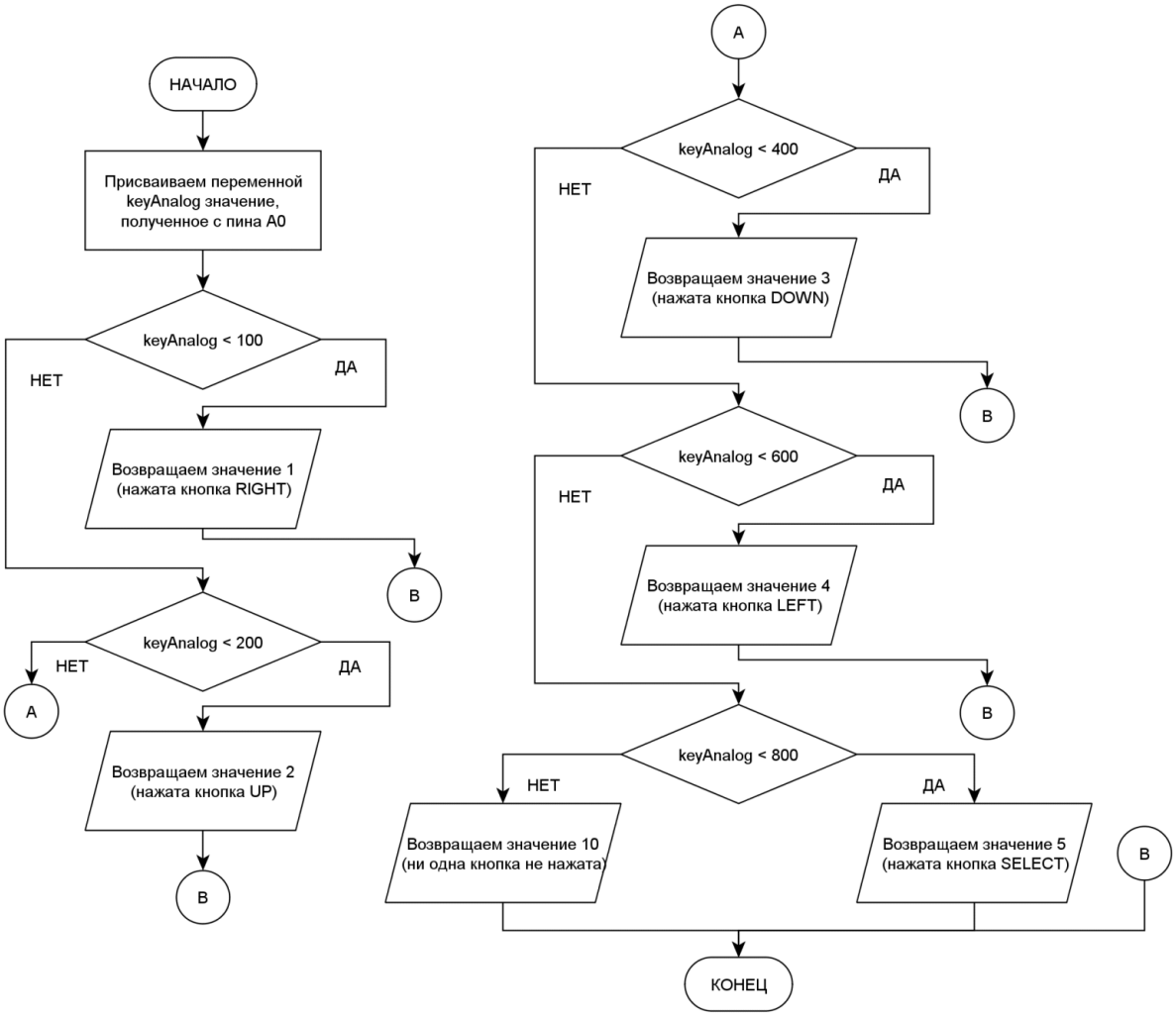

Рис. 2.3 Блок-схема функции clickButton

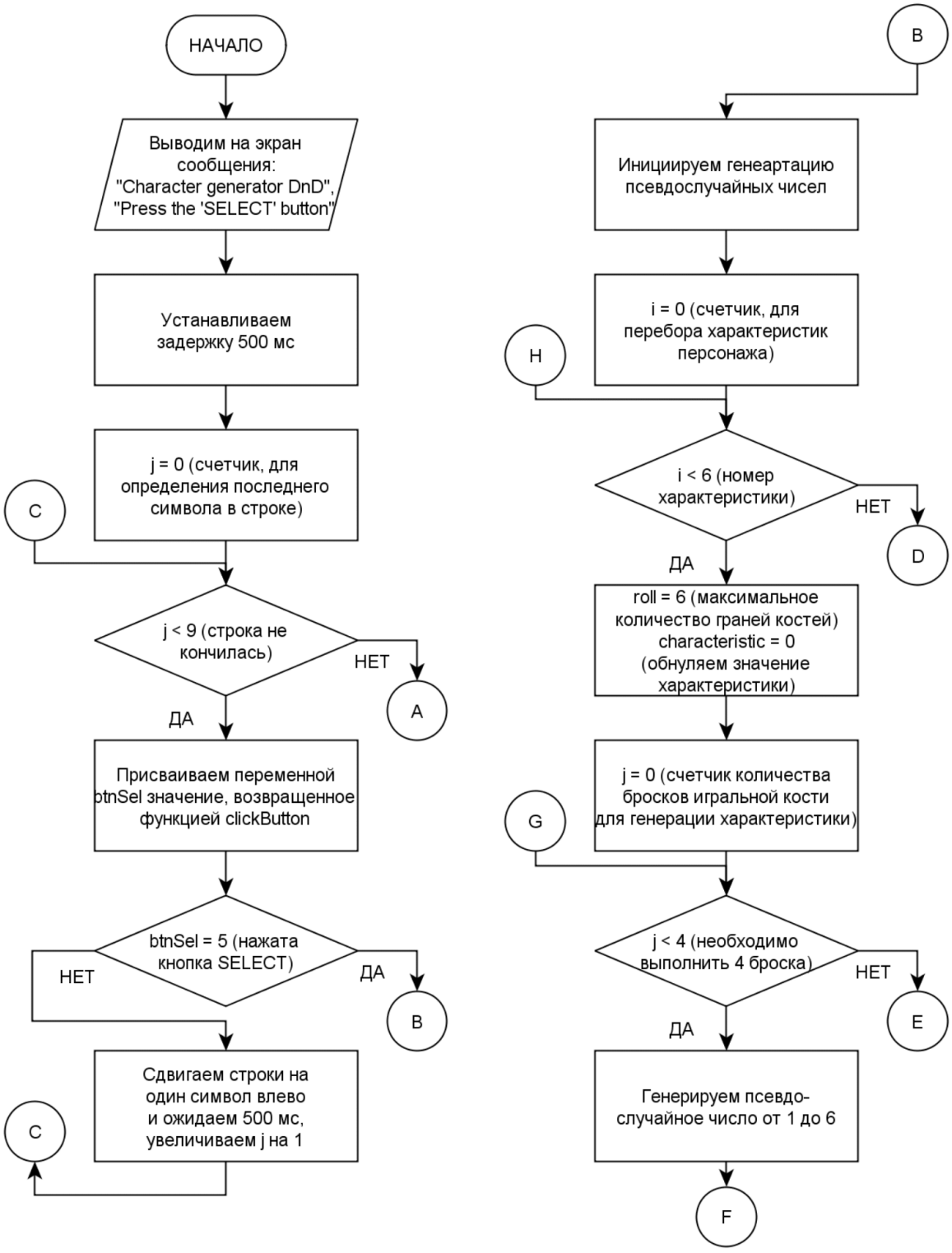

Рис. 2.4 Блок-схема функции mainText

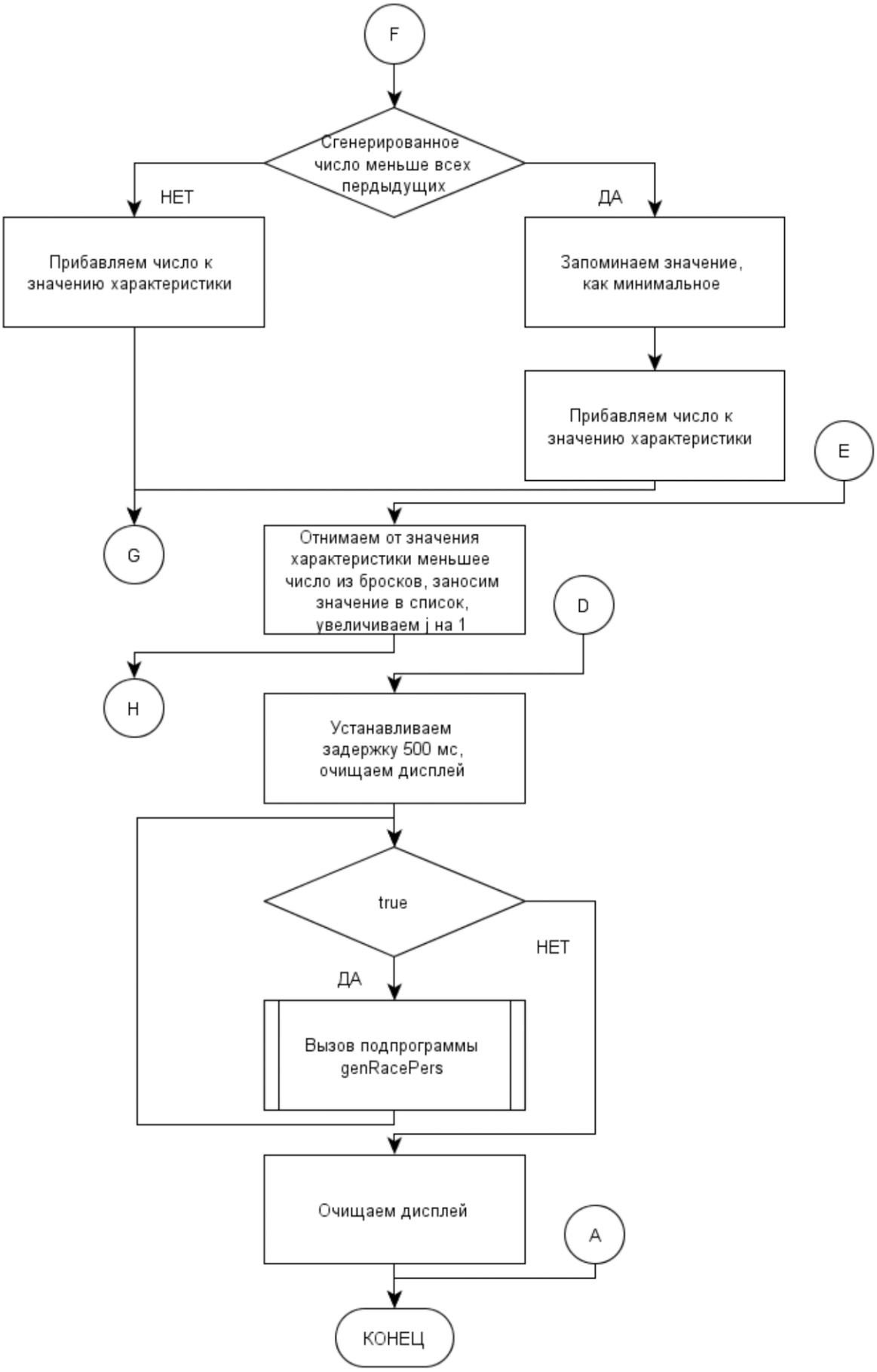

Рис. 2.5 Блок-схема функции mainText

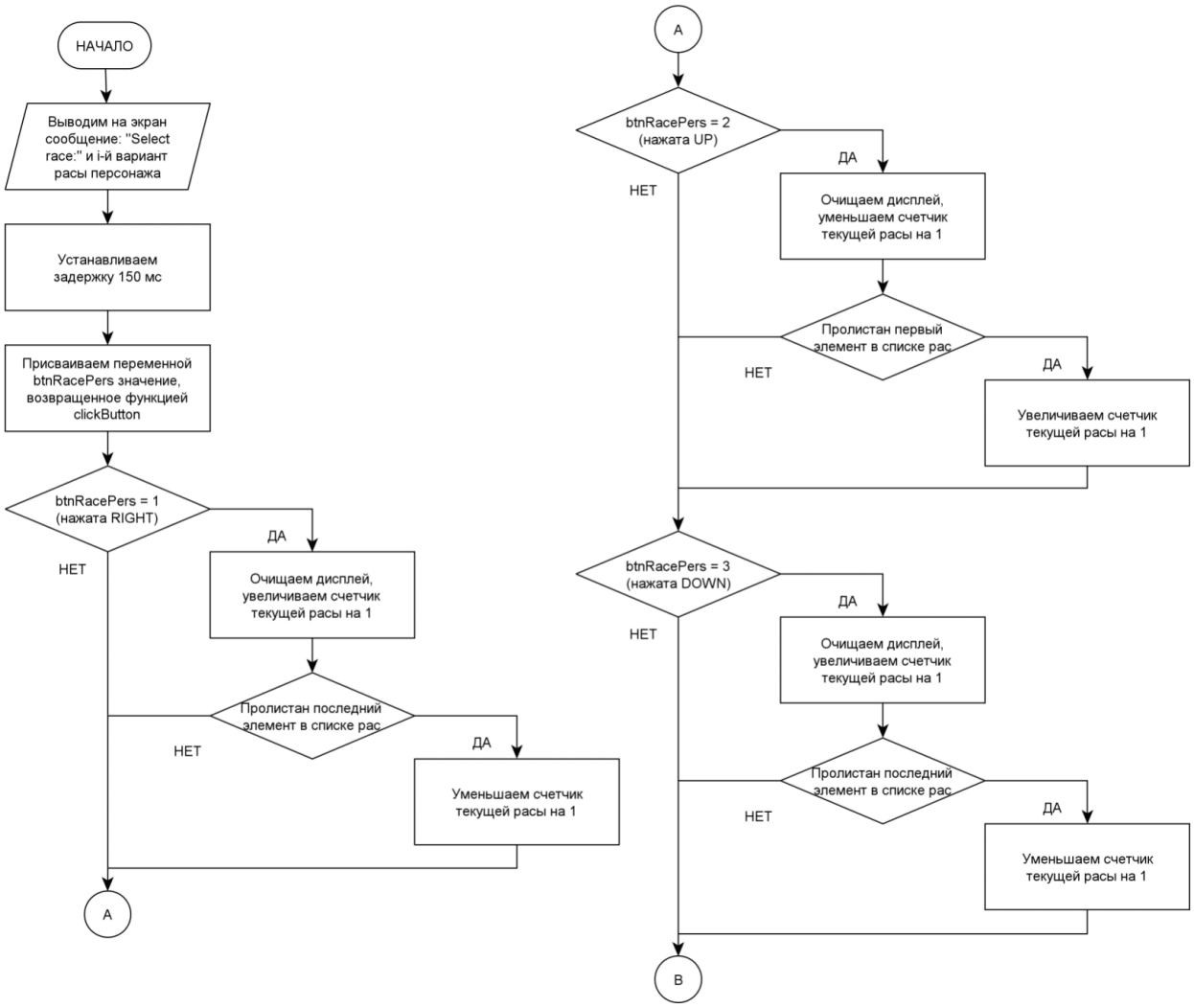

Рис. 2.6 Блок-схема функции genRacePers

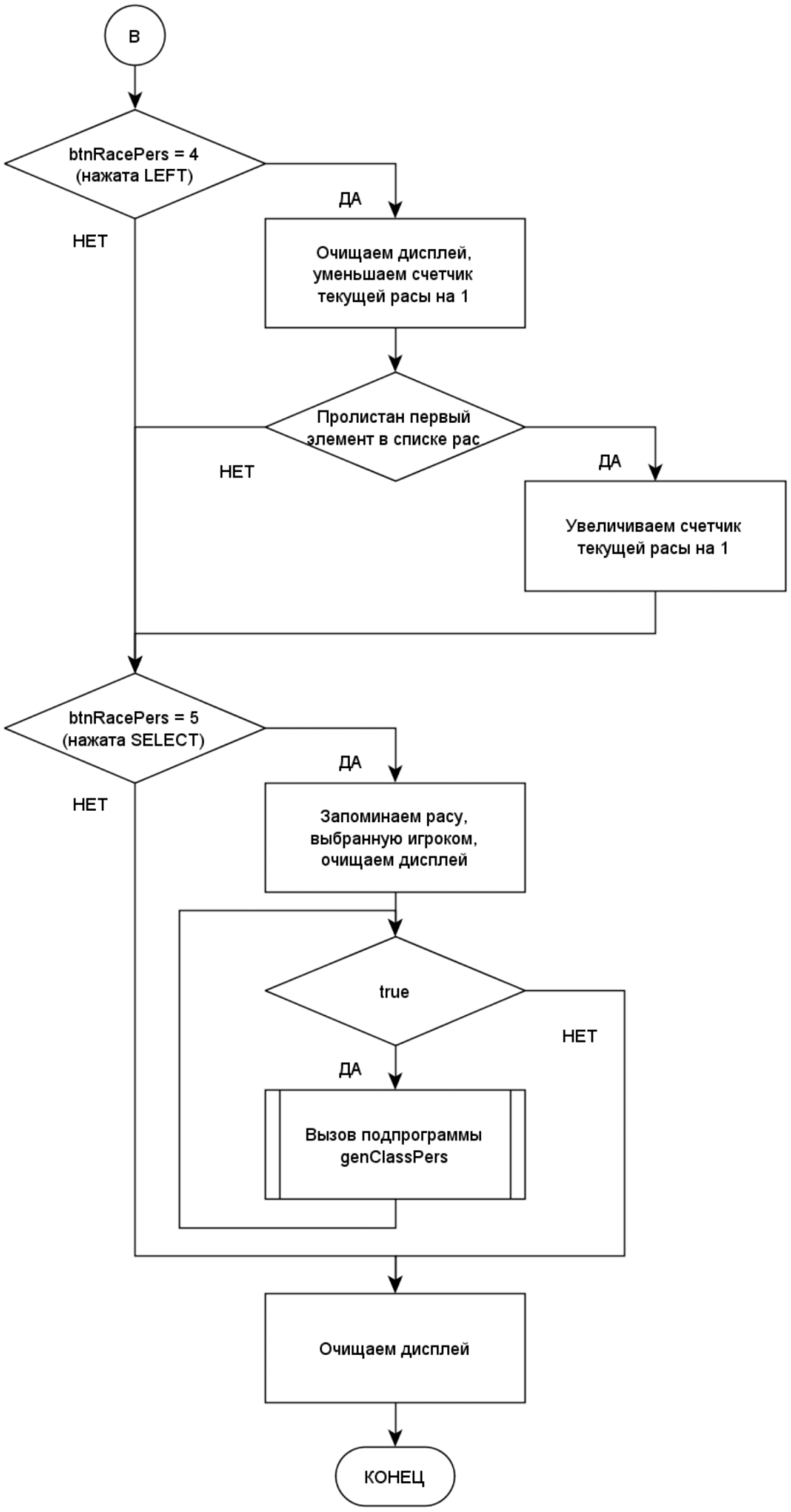

Рис. 2.7 Блок-схема функции genRacePers

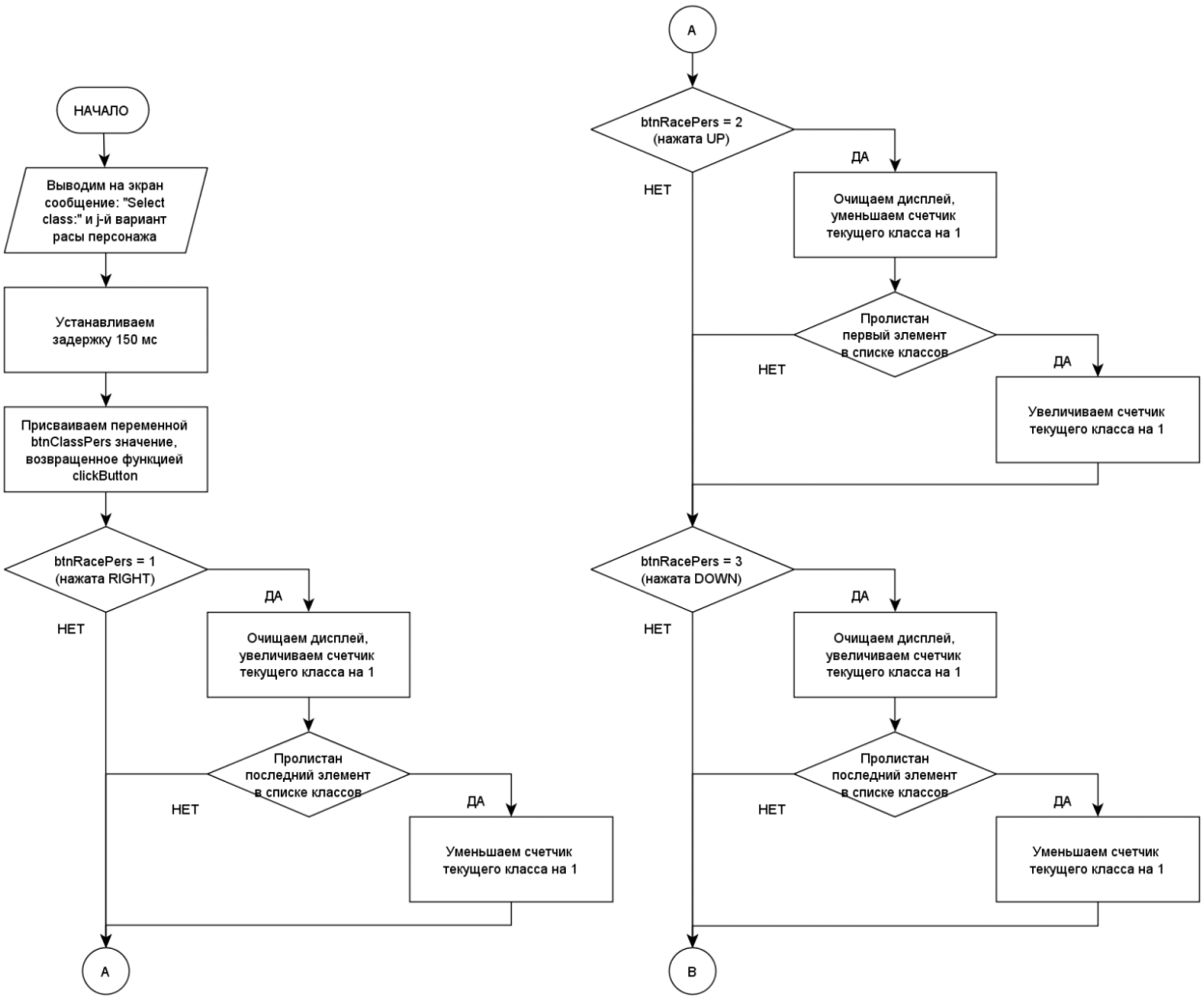

Рис. 2.8 Блок-схема функции genClassPers

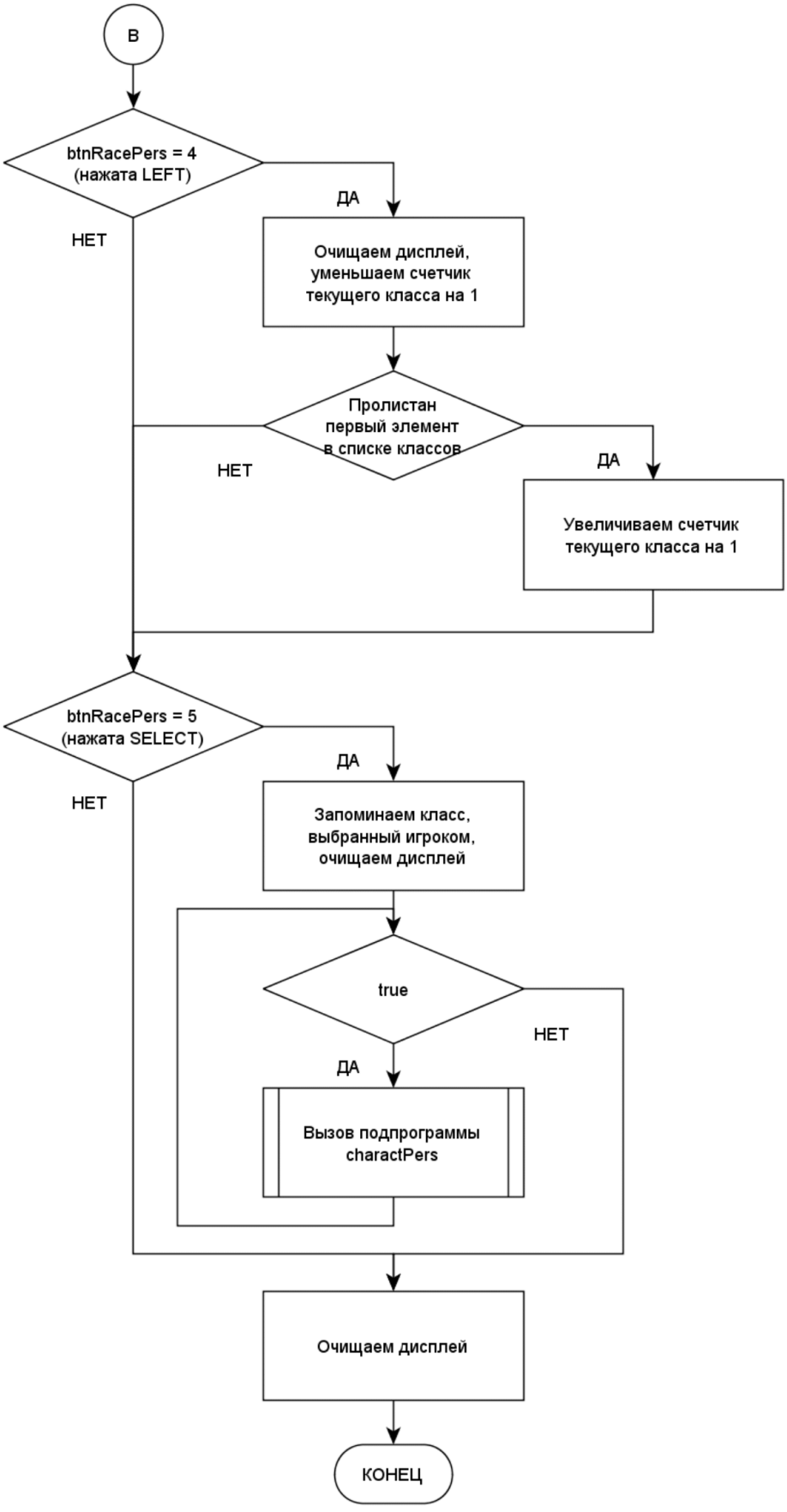

Рис. 2.9 Блок-схема функции genClassPers

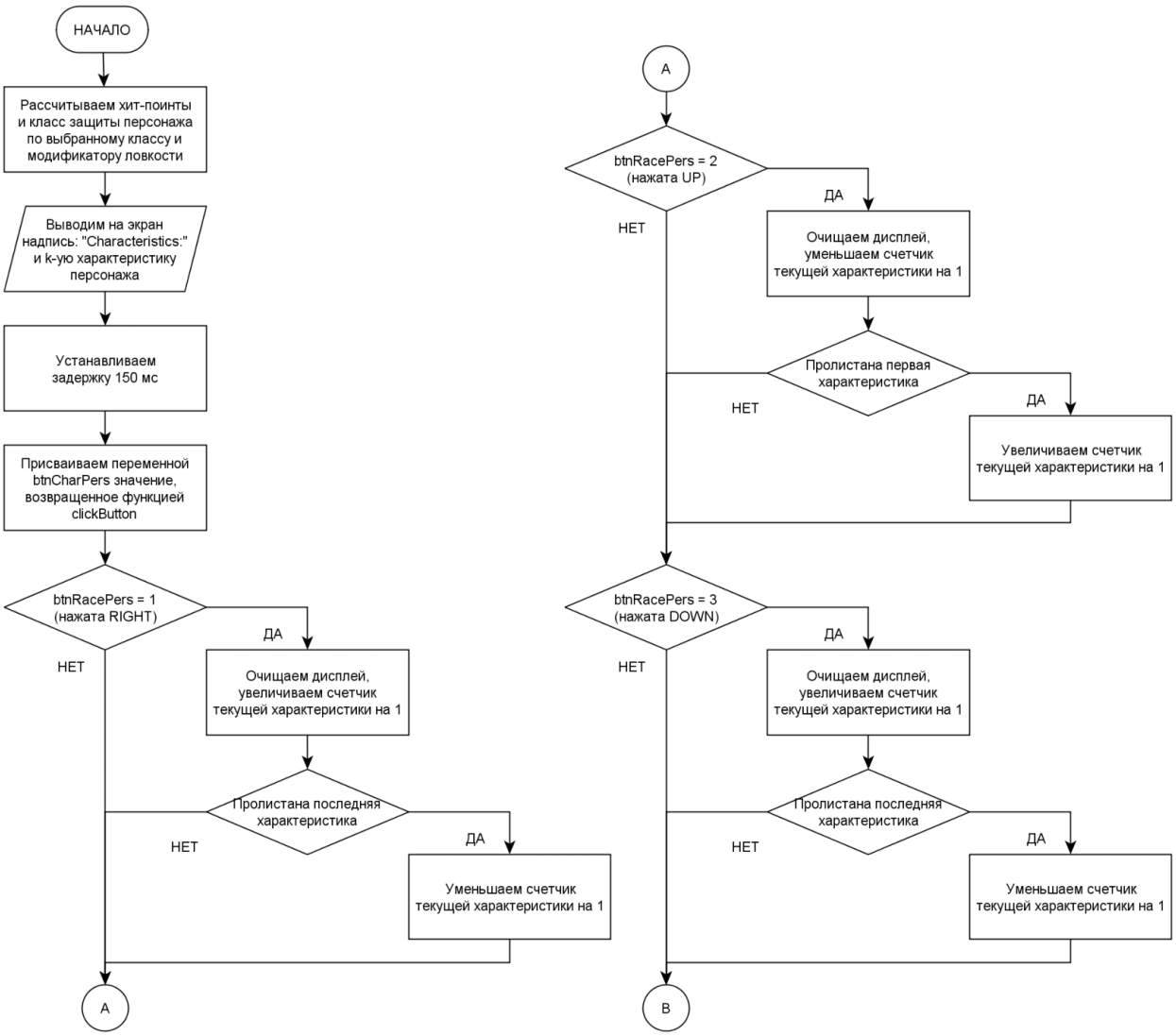

Рис. 2.10 Блок-схема функции charactPers

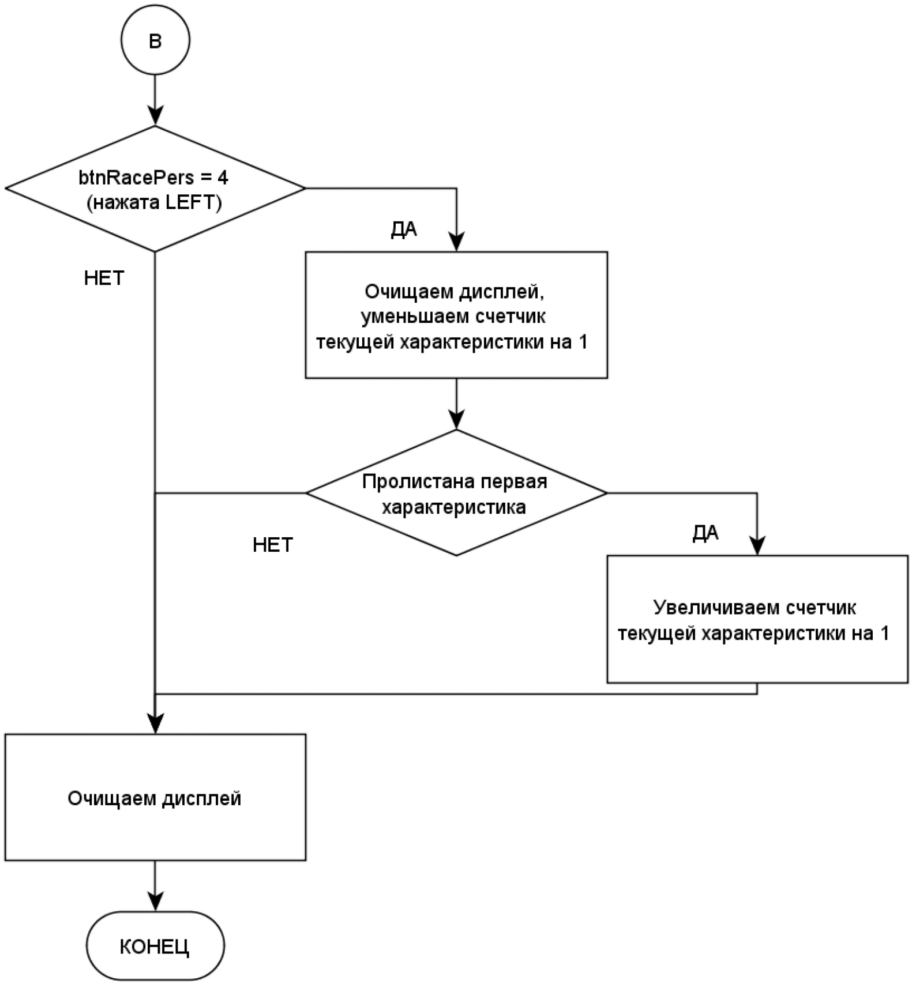

Рис. 2.11 Блок-схема функции charactPers