

Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №12 «Компьютерные системы и технологии»



# ОТЧЕТ

**О выполнении лабораторной работы №1  
Алгоритмизация обработки целых чисел.**

**Студент:** Пахомов А.К

**Группа:** Б22554

**Преподаватель:** Доценти́ков Ю. Б.

Москва — 2022

# 1. Формулировка индивидуального задания

Вариант №3. Дано целое число. Получить новое число только из разных цифр введенного числа.  
Например из числа 23241 необходимо получить число 2341.

## 2. Описание использованных типов данных

При выполнении данной лабораторной работы использовался встроенный тип данных `Int`, предназначенный для работы с целыми числами.

## 3. Описание использованного алгоритма

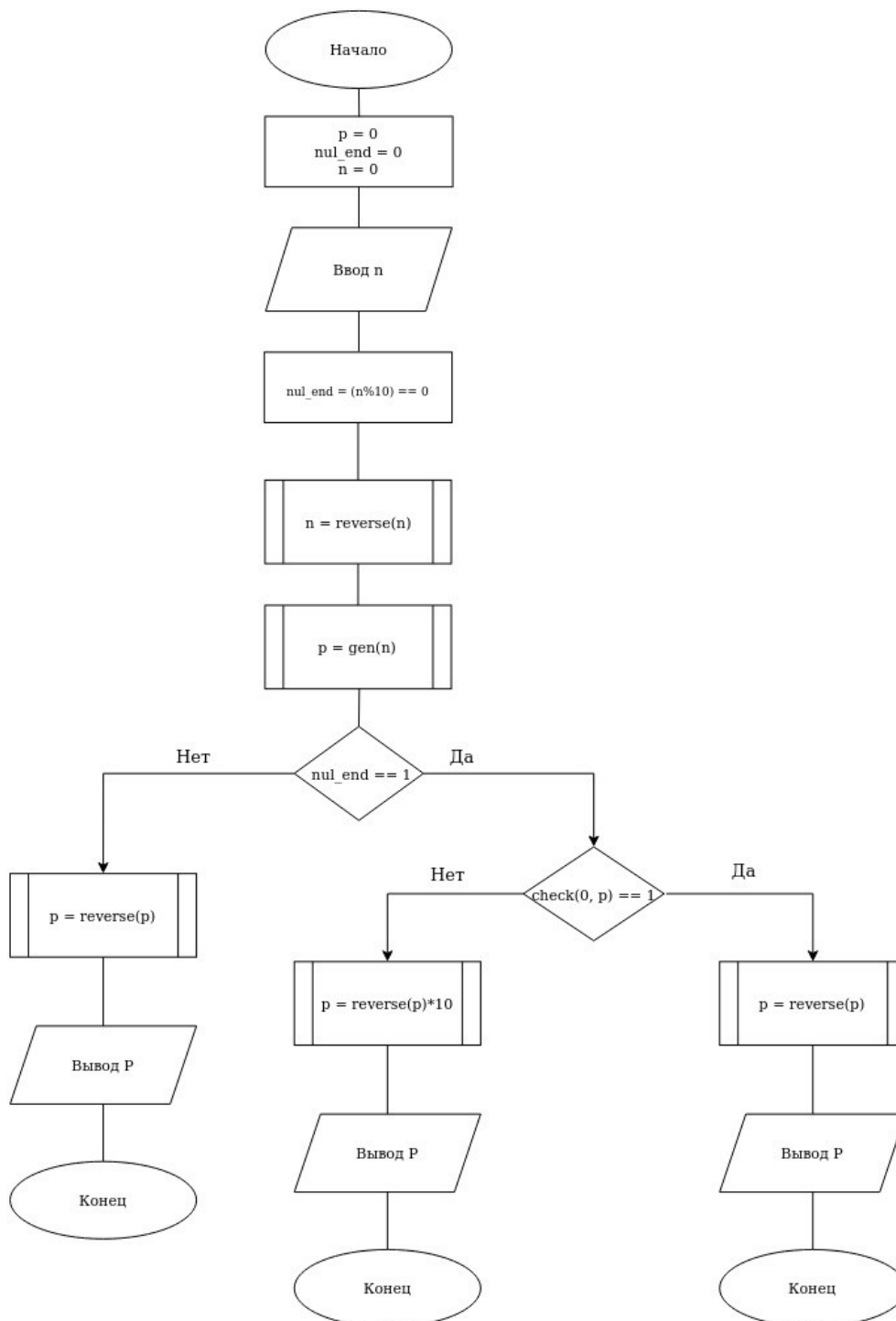


Рис. 1: Блоксхема алгоритма работы функции `main()`

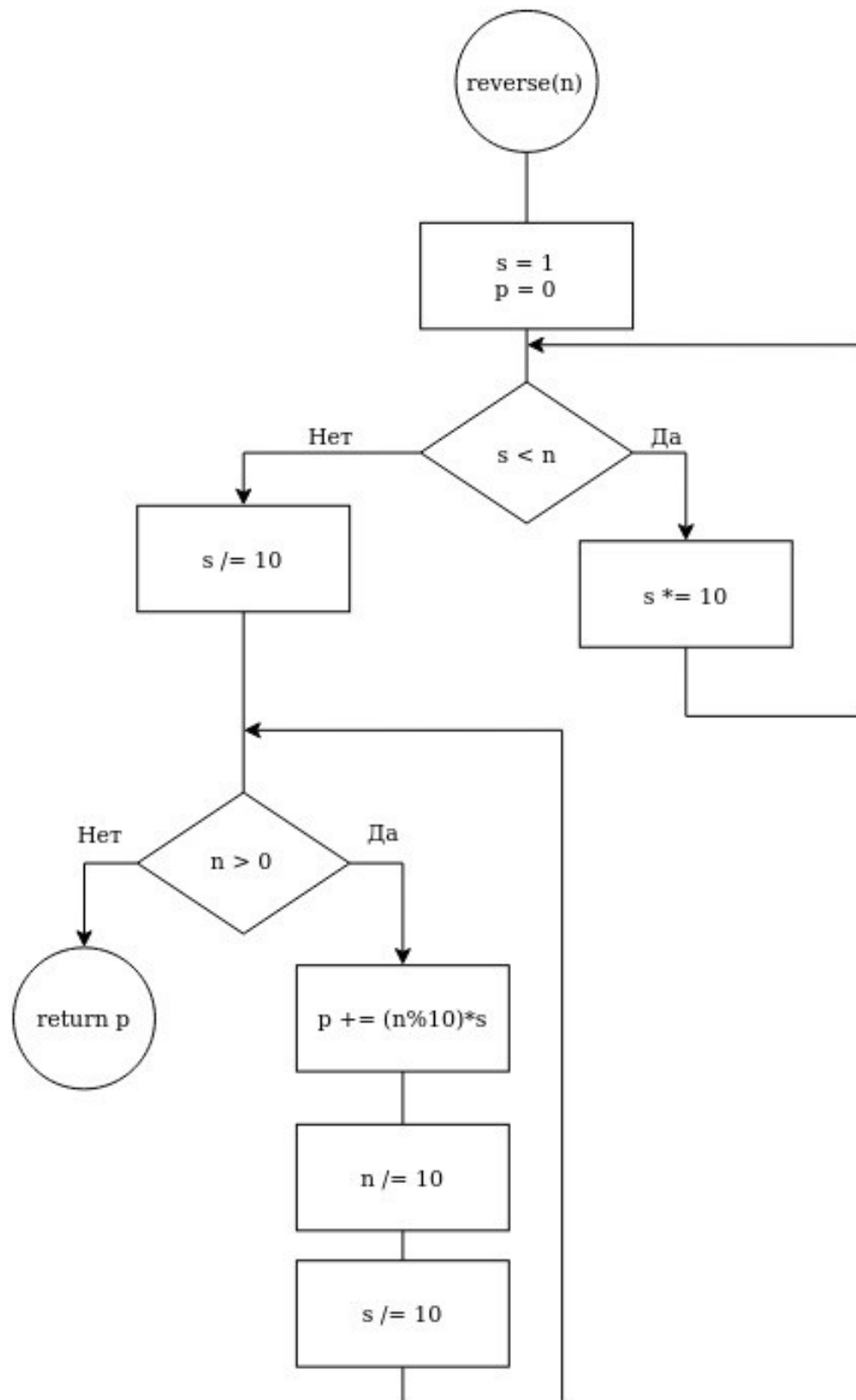


Рис. 2: Блоксхема алгоритма работы функции `reverse()`

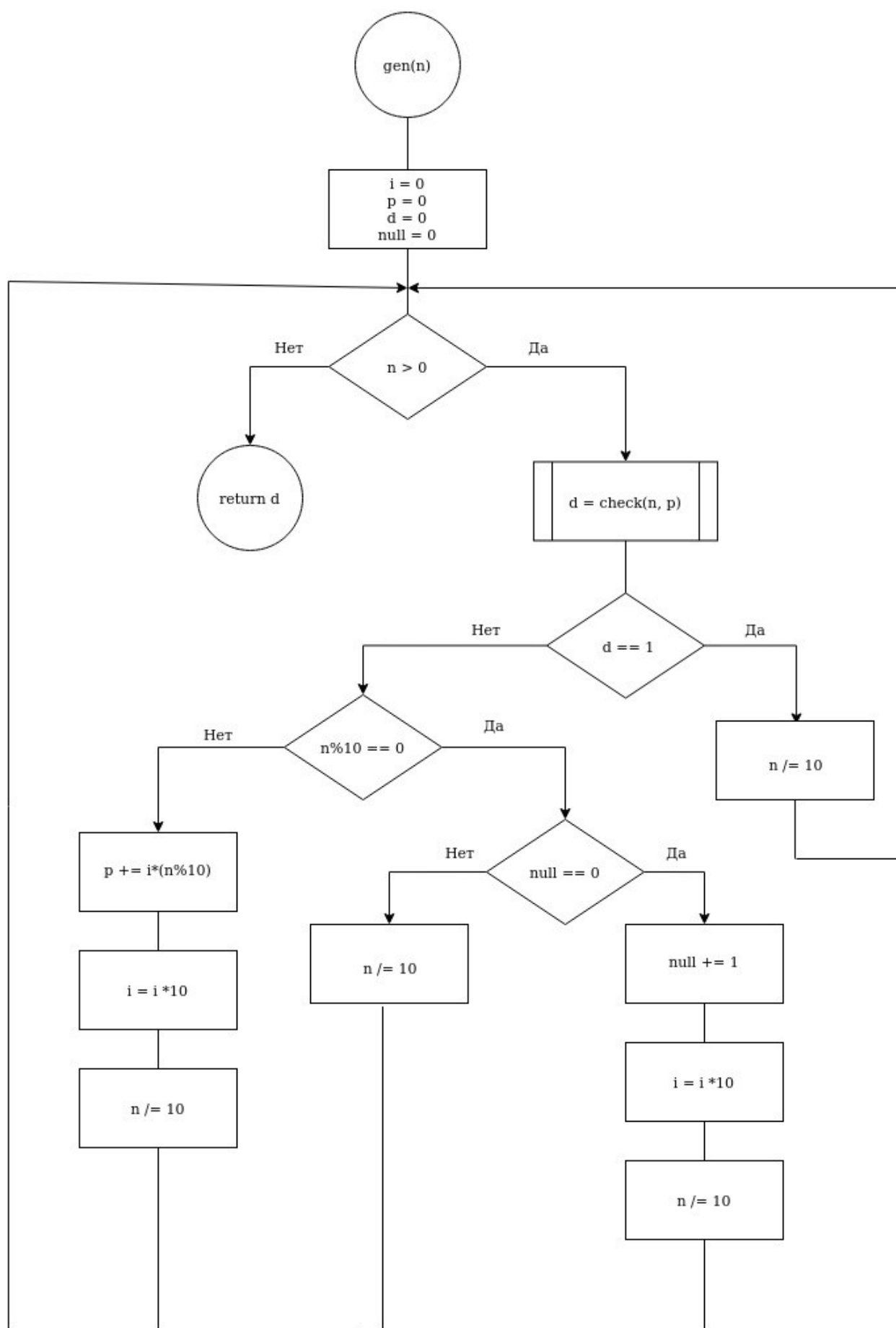


Рис. 3: Блоксхема алгоритма работы функции `gen()`

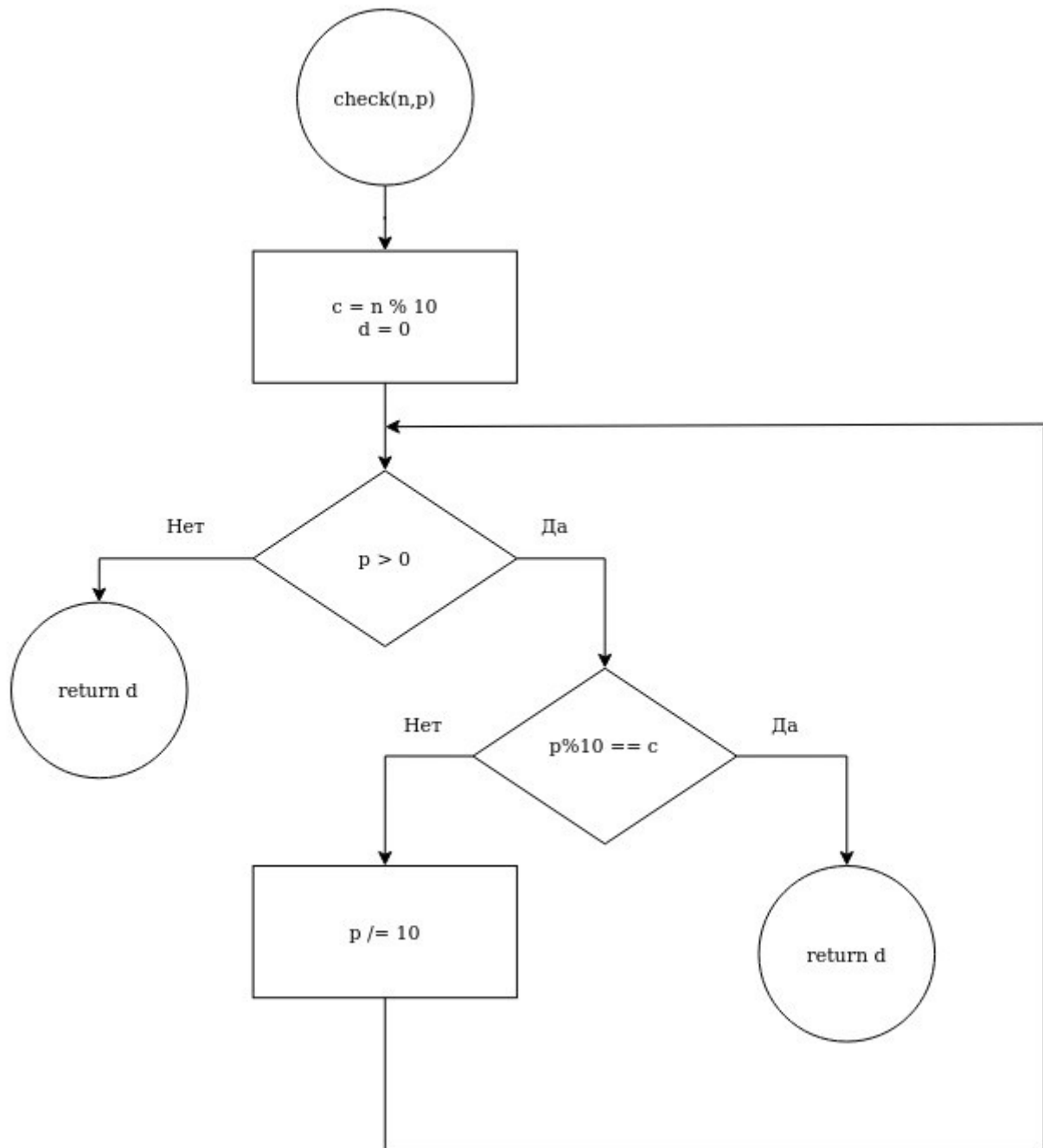


Рис. 4: Блоксхема алгоритма работы функции `check()`

## 4. Исходные коды разработанных программ

```
#include<stdio.h>

int reverse(int n){
    int p = 0;
    int s = 1;

    while (s < n){ /*Узнаем число, которое поможет расставлять цифры в нужные разряды */
        s *= 10;
    }
    s /= 10; /*В прошлом цикле получили число в 10 раз больше чем нам надо, так что делим */
    while (n > 0){ /* Берем последнюю цифру и при помощи числа S, которое мы узнали ранее, начинаем
собирать новое, т.е. перевернутое число*/
        p += (n%10)*s;
        n /= 10;
        s /= 10;
    }
    return p;
}

int check(int n, int p){ /* Функция проверки наличия цифры в числе */
    int c = n % 10;
    int d = 0;
    while (p > 0){
        if (p%10 == c){
            d = 1;
            return d;
        }
        else{
            p /= 10;
        }
    }
    return d;
}

int gen(int n){
    int i = 1; /* Снова число, на которое умножаем, чтобы поставить цифру в нужный разряд */
    int nul = 0; /* Переменная, которая показывает, есть ли в новом числе ноль или нет */
    int p = 0; /* Новое число */
    int d = 0; /* Переменная флаг, которая используется далее, для того чтобы понять, повторилась ли
цифра */
    while (n > 0){
        d = check(n, p);
        if (d == 1){
            n /= 10;
        }
        else{
            if (n%10 == 0){ /* Проверяем, является ли новая цифра нулем */
                if (nul == 0){ /* Проверяем наличие нулей в новом числе */
                    nul += 1;
                    i *= 10;
                    n /= 10;
                }
                else{
                    n /= 10;
                }
            }
        }
    }
}
```

```

    }
    else{
        p += (n%10)*i;
        i *= 10;
        n /= 10;
    }
}
return p;
}

int main(){
    int n = 0; /* Число, мы вводим */
    int nul_end = 0; /* Переменная которая показывает, есть ли ноль на конце у числа */
    printf("Enter the number:");
    scanf("%d", &n); /* Ввод числа */
    nul_end = (n%10 == 0); /* Проверяем, есть ли ноль на конце */
    n = reverse(n); /* Так как надо убрать все повторы цифр, начиная с левого конца,
переворачиваем число */
    int p = gen(n); /* Генерируем новое число, которое не содержит дубликатов */
    if (nul_end == 1){ /* Если в начальном числе на конце был ноль, то проверяем, есть
ли в новом числе ноль */
        if (check(0, p) == 1){
            printf("%d \n", reverse(p)); /* Если есть, просто выводим число */
        }
        else{
            printf("%d \n", reverse(p)*10); /* Если нет, то "Прерисовываем" нолик
в конце */
        }
    }
    else{
        printf("%d \n", reverse(p)); /* Если нуля на конце не было, то просто выводим
число */
    }
    return 0;
}

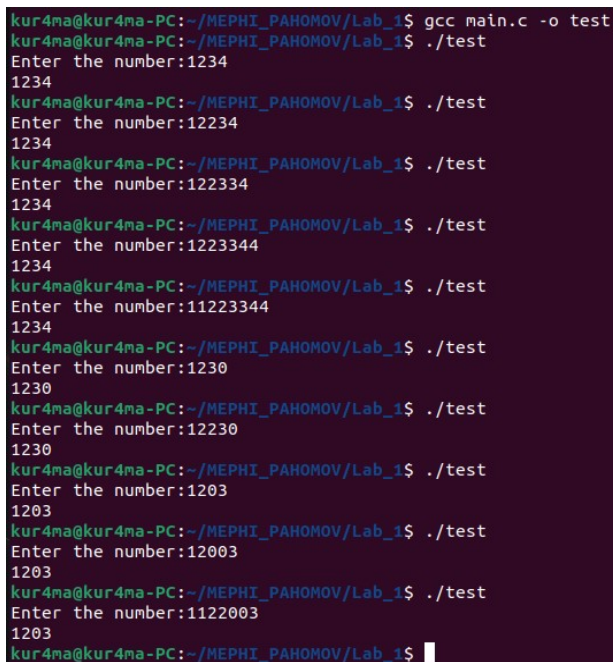
```

## 5. Описание тестовых примеров

Таблица 1: Тестовые примеры

| Значение n | Ожидаемое значение p | Полученное значение p |
|------------|----------------------|-----------------------|
| 1234       | 1234                 | 1234                  |
| 12234      | 1234                 | 1234                  |
| 122334     | 1234                 | 1234                  |
| 1223344    | 1234                 | 1234                  |
| 11223344   | 1234                 | 1234                  |
| 1230       | 1230                 | 1230                  |
| 12230      | 1230                 | 1230                  |
| 1203       | 1203                 | 1203                  |
| 12003      | 1203                 | 1203                  |
| 1122003    | 1203                 | 1203                  |

## 6. Скриншоты



```
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ gcc main.c -o test
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:1234
1234
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:12234
1234
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:122334
1234
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:11223344
1234
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:11223344
1234
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:1230
1230
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:12230
1230
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:1203
1203
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:12003
1203
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$ ./test
Enter the number:1122003
1203
kur4ma@kur4ma-PC:~/МЕРНИ_ПАНОМОВ/Lab_1$
```

Рис. 5: Сборка и запуск программы test

## 7. Выводы

В ходе выполнения данной работы на примере программы, убирающей повторяющиеся цифры из числа, были рассмотрены базовые принципы работы построения программ на языке С и обработки целых чисел:

1. Организация ввода/вывода.
2. Разработка функций.
3. Объявление и использование переменных.
4. Выполнение простейших арифметических операций над целочисленными операндами.