|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация (РК)

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКИ**

Студент Камалов Антон Павлович

Группа РК6-25Б

Модуль № 1

Вариант H08

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Камалов А.П.**

*подпись, дата фамилия, и.о.*

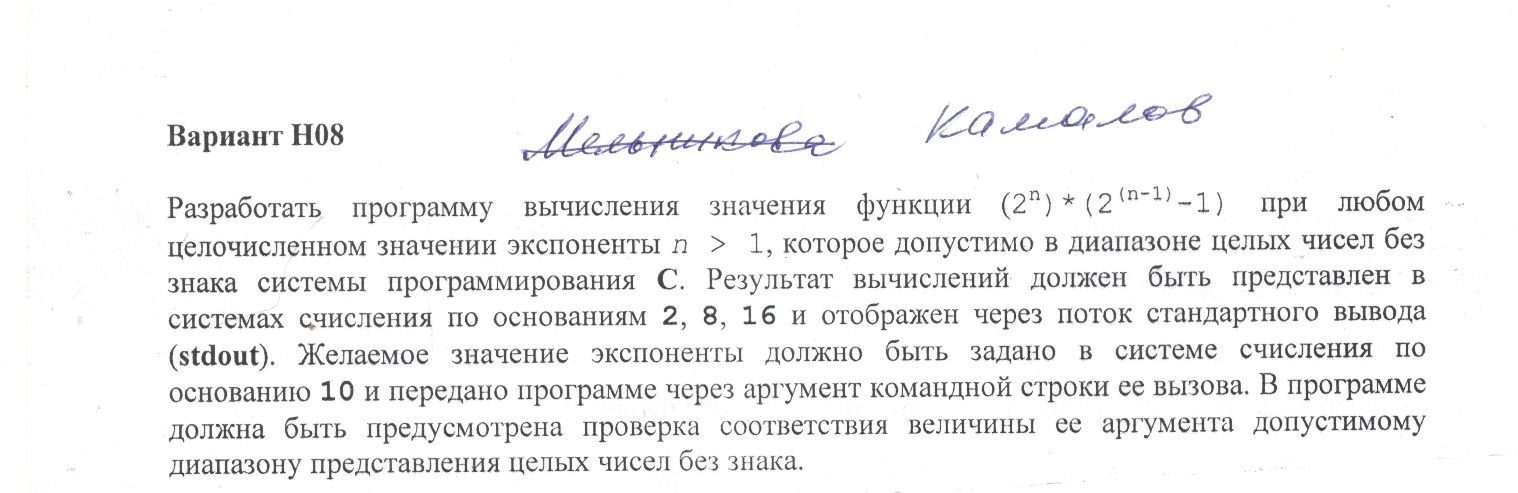
Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Волосатова Т.М.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2023 г.*

**Задание**



**Описание программы**

**Входные данные:** целочисленная экспонента, введённая из командной строки (***long*** n ∈ [2;32])

**Выходные данные:** представление значения функции в двоичной, восьмеричной и шестнадцатеричной системе счисления.

**Общий алгоритм решения:**

1. Проверяем количество аргументов командной строки. Если их меньше 2, то программа завершает работу с кодом возврата -1;
2. Проверяем правильность ввода числа. Для этого считываем посимвольно нашу строку с предполагаемым числом. Если хотя бы один из символов не число или знаки '+', '-' стоит не в начале числа, то программа завершает работу с кодом возврата -2;
3. Создаём переменную **n** типа ***long***, которая будет хранить в себе значение экспоненты, полученной из аргумента командной строки.
4. Так как мы хотим хранить максимально возможное значение функции, то результат должен быть представлен с типом ***unsigned long long int*** (значение функции из-за условия задания всегда больше 0).

Чтобы выяснить максимально возможную экспоненту нужно решить следующее неравенство:

,

где – максимальное значение для беззнакового типа ***long long int***.

Решив его, получаем:

Следовательно, выполняем проверку переменной n на принадлежность отрезку [2;32]. В противном случае программа завершает работу с кодом возврата -3;

1. Создаём переменную **result** типа ***unsigned long long int***, которая равняется значению исходной функции. Чтобы возвести 2 в n-ую степень, то используем функцию **pow2()**, которая реализует побитовый сдвиг числа 1 на n позиций, чтобы получить результат .
2. Выводим представления переменной **result** в двоичной, восьмеричной и шестнадцатеричной системе счисления.

Число в шестнадцатеричной и восьмеричной системе счисления выводится в поток стандартного вывода с помощью функции **printf()** со спецификаторами «**llo»** и «**llx**», то вывод result в двоичной СС осуществляется с помощью рекурсивной функции **print\_binary()** – если число при сдвиге справа на 1 не равно 0, то в функция снова вызывается, но её аргументом будут является число, сдвинутое на 1 бит. В противном случае мы отображаем 1, если бит в **number** присутствует, или 0.

**Список литературы**

1. Керниган Б., Ритчи Д. Язык программирования C: пер. с англ / под ред. В.Л. Бродового. М.: Издательство Вильямс, 2019. 288 с. [Brian Kernighan, Dennis Ritchie. The C Programming Language. 2nd ed. Prentice Hall, 1988. 288 p.].
2. Федорук Е.В., Шварц Д.Т. Основы программирования. Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=OP2/OP\_T.cou (дата обращения 05.04.2023).

**Приложение 1**

Код программы

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#include <ctype.h>

int is\_number(char str[]) {

int i = 1, flag = 0;

while (str[i]) {

if ((str[i] == '+' || str[i] == '-') && i != 0) return 0;

else if (!isdigit(str[i])) return 0;

i++;

}

return 1;

}

unsigned int pow2(unsigned int n) {

return 1 << n;

}

void print\_binary(unsigned int number) {

if (number >> 1) {

print\_binary(number >> 1);

}

printf("%c", (number & 1) ? '1' : '0');

}

int main(int argc, char \*\*argv){

if (argc < 2){

printf("No argument!\n");

return -1;

}

if (!is\_number(argv[1])){

printf("Argument is not integer number!\n");

return -2;

}

long n = strtol(argv[1],NULL,10);

if (n > 32 || n < 2){

printf("Exponent is not correct!\n");

return -3;

}

unsigned long long result = pow2(n) \* (pow2(n-1) - 1);

printf("Binary : "), print\_binary(result);

putchar('\n');

printf("Octal : %llo\n", result);

printf("Hex : %llx\n", result);

}

**Приложение 2**

Результат работы программы

