|  |  |
| --- | --- |
| Изображение выглядит как текст, эмблема, герб, нашивка  Автоматически созданное описание | **Министерство образования и науки 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 Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

Дисциплина Методы комбинаторных вычислений

Название работы Лабораторная работа №4

Студент Долженко Анастасия Тимофеевна

Группа РК6-52Б

Вариант 7

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Долженко А.Т.**

*подпись, дата фамилия, и.о.*

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

*подпись, дата фамилия, и.о.*

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

*Москва, 2024 г.*

**Оглавление**

[Задание 3](#_Toc178644053)

[Алгоритм работы программы 3](#_Toc178644054)

[Приложение 1. Код программы 5](#_Toc178644055)

[Приложение 2. Результаты работы программы 6](#_Toc178644056)

[Список литературы 6](#_Toc178644057)

# **Задание**

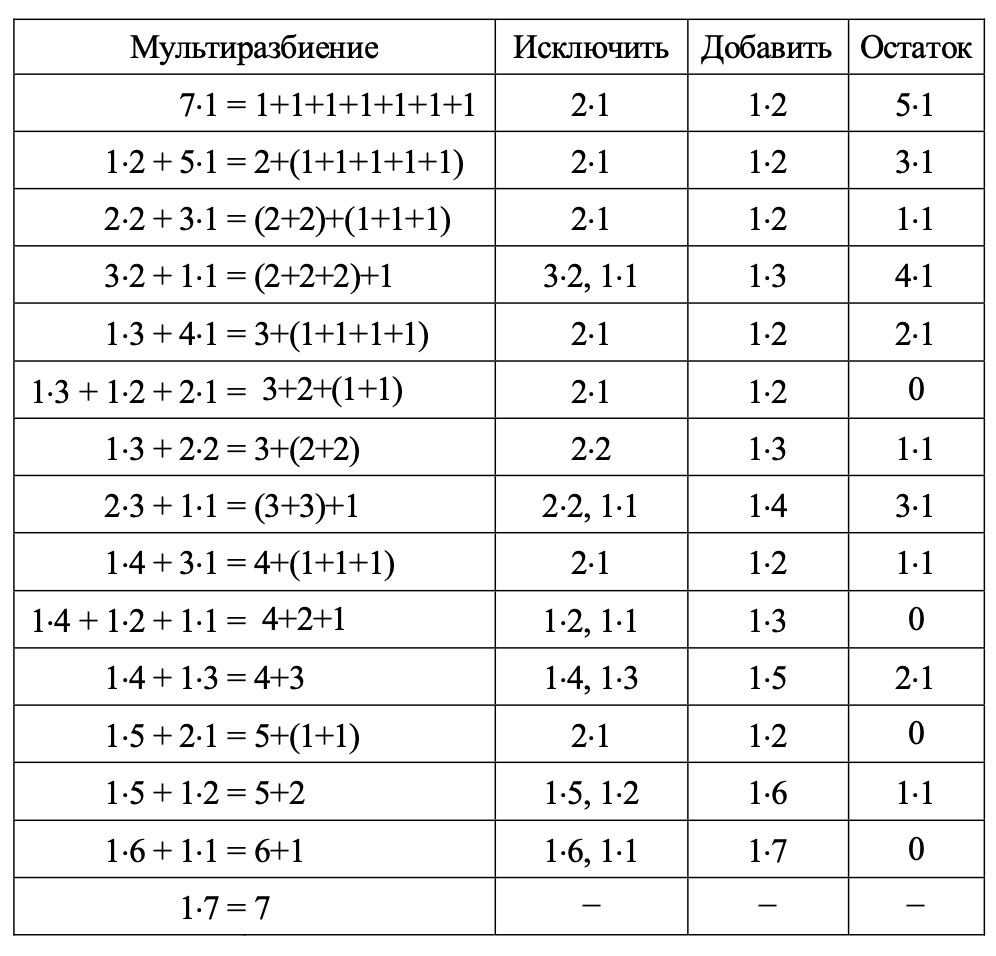
Вариант 7

Изображение выглядит как текст, Шрифт, информация

Автоматически созданное описание

Вариант 7: Перечислить все разбиения заданного целого числа n>0 на слагаемые, где максимальное из них не входит в заданный интервал [L;U]. Для генерации разбиений следует применить алгоритм Эрлиха с мульти записью слагаемых каждого разбиения в порядке убывания своих величин слева направо и знаком + между ними.

# **Алгоритм работы программы**

1. *Алгоритм Эрлиха* порождает все разбиения любого заданного числа в словарном порядке, используя мультимножественную запись слагаемых, где учитывается их кратность. Это дает возможность записать разбиение любого числа *n* на *m < n* мультислагаемых в виде
2. В этой записи обозначает кратность слагаемого , а все слагаемые перечислены в порядке убывания их величин:
3. Например, разбиение числа 7 на 2 мультислагаемых 2 и 1 с кратностями 2 и 3, соответственно, выглядит так:
4. Мультимножественная запись слагаемых позволяет построить алгоритм генерации разбиений целых чисел, который переходит от текущего разбиения к следующему в словарном порядке, рассматривая только самое правое мультислагаемое без анализа предыдущих частей. Способ конструирования очередного разбиения зависит от кратности последнего мультислагаемого текущего разбиения. При этом требуется рассмотреть следующие два случая:
5. В первом — исключить из разбиения значение (), чтобы получить возможность добавить еще одно мультислагаемое со значе- нием ( + 1). Во втором — добавить еще одно мультислагаемое со значением (− 1 + 1), исключив два последних. В любом случае то, что остается от уменьшаемых мультислагаемых должно быть превращено в соответствующее число единиц для последнего мультислагаемого следующего разбиения. Этот процесс должен начинаться с разбиения, где все части равны единице, и завершаться разбиением из одного мультислагаемого. Выполнение шагов алгоритма Эрлиха, где показана генерация разбиений числа 7, иллюстрируется ниже:
6. 

# **Приложение 1. Код программы**

#include <stdio.h>

#include <stdlib.h>

// Поиск максимального слагаемого

int maxsum(int\* p, int n) {

int max = 0;

for (int i=0; i<n; i++)

if(p[i] != 0 && max < i) max = i;

return max+1;

}

// Печать разбиения в формате суммы

void print(int\* p, int n) {

for (int i=n-1; i>-1; i--)

if (p[i] != 0)

for (int j=0; j<p[i]; j++)

printf("%d+", i+1);

printf("\b \n");

}

// Алгоритм Эрлиха

int\* iter(int\* p, int n) {

int m, left, j;

for (m=0; m<n-1; m++)

if (p[m] != 0) {

left = 0;

if (p[m] == 1) {

for (j=m+1; !p[j] && j<n; j++);

left += (m+1)\*p[m] + (j+1)\*p[j];

p[m] = 0;

p[j] = 0;

m = j;

} else {

p[m] -= 2;

left += (m+1)\*2;

}

p[m+1] += 1;

left -= m+2;

p[0] += left;

return p;

}

return 0;

}

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

int n = 7, l = 8, u = 9, temp;

int\* p = (int\*) malloc(n \* sizeof(int));

if (argc > 1) {

temp = atoi(argv[1]);

if (temp > 0) {

n = temp;

} else {

printf("|n| должно быть целым числом больше нуля\n");

free(p);

return 1;

}

}

if (argc > 2) {

temp = atoi(argv[2]);

if (abs(temp) <= n) {

l = temp;

} else {

printf("|l| должно быть целым числом и l <= n\n");

free(p);

return 1;

}

}

if (argc > 3) {

temp = atoi(argv[3]);

if (abs(temp) <= n) {

u = temp;

} else {

printf("|u| должно быть целым числом и l <= u <= n\n");

free(p);

return 1;

}

}

for (int i=0; i<n; i++)

p[i] = 0;

p[0] = n;

do if (maxsum(p, n) < l || maxsum(p, n) > u) print(p, n);

while ((iter(p, n)));

free(p);

return 0;

}

# **Приложение 2. Результаты работы программы**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

# **Список литературы**

1. Методы комбинаторных вычислений – Т.М. Волосатова, С.В. Родионов, 2011
2. http://bigor.bmstu.ru/?cnt/?doc=Comby/base.cou