Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

за 3 семестр

По дисциплине: «Дискретная математика»

Тема: «Множества»

Выполнил:

Студент 2 курса

Группы ПО-6(1)

Мартынович Д. М.

Проверил:

Глущенко Т. А.

2021

Цель: изучить базовые понятия теории множеств и операции над множествами.

Вариант 10

На универсуме U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11} заданы множества:

А = {1, 3, 5, 6, 7, 8} В = {2, 3, 4, 5, 8} С = {1, 2, 3, 5, 6}

Выражение: (А ⋃ ( В **∩** С).

Ход работы:

1. Для заданного множества *A* построить *булеан*.

void t1(int arr[], int size)//построить булеан множества А

{

cout << "1" << endl;

cout << "Булеан множетсва." << endl;

cout << "{{ ";

for (int i = 0; i < pow(2, size); i++) {

cout << " {";

for (int j = 0, num = i; num != 0; j++) {

if (num % 2 == 1) {

cout << arr[j] << ", ";

}

num /= 2;

}

cout << '\b' << '\b' << "},";

if (i != pow(2, size) - 1) {

cout << endl;

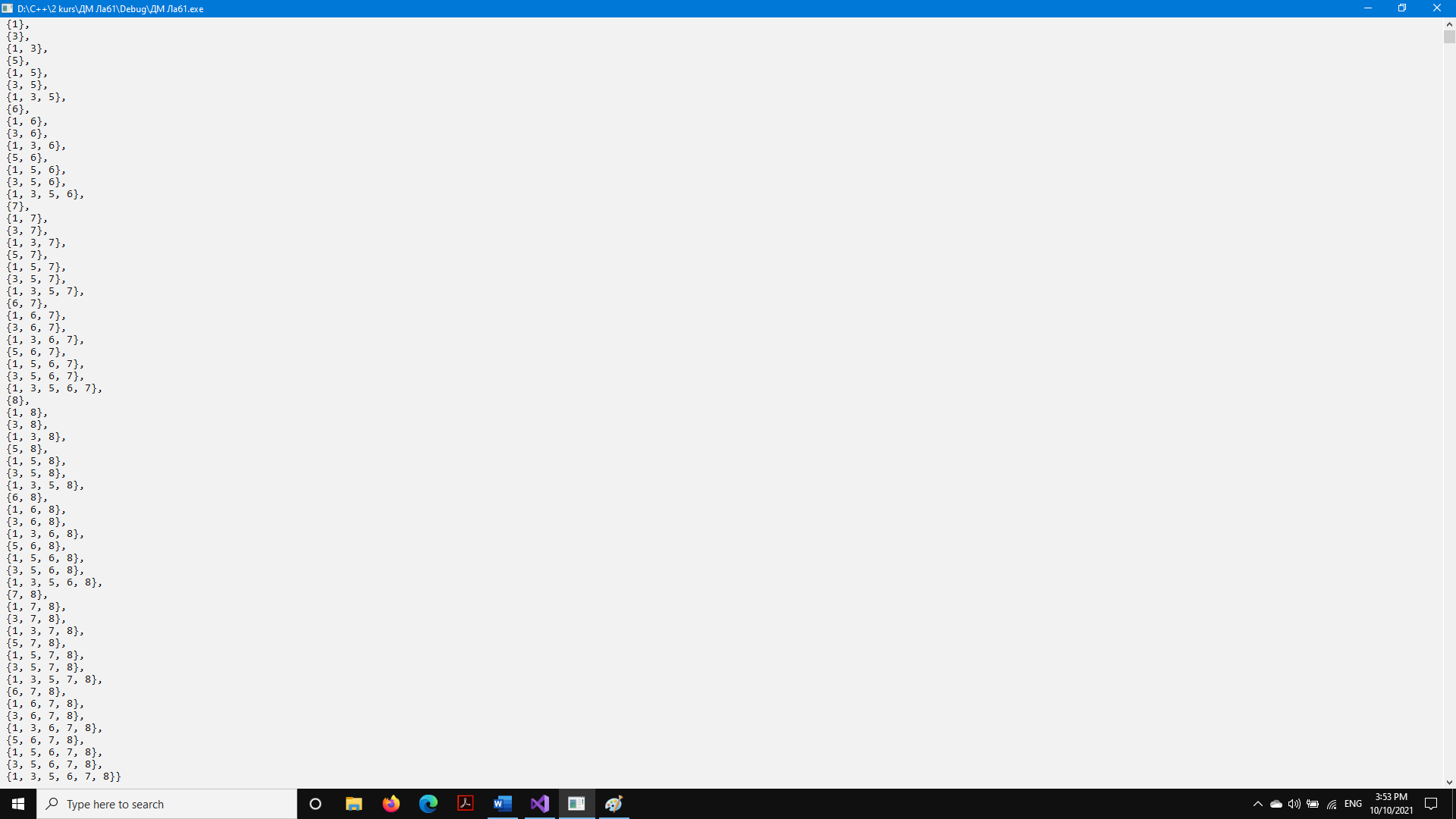
}

}

cout << '\b' << "}";

cout << endl;

}



1. На выбор реализовать сортировку *слиянием* либо алгоритмом *«слияние»* построить объединение множеств *A* и *B*.

void t2(int A[], int size\_A, int B[], int size\_B) {

cout << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "2" << endl;

cout << "Сортировка слиянием." << endl;

int\* arr = new int[size\_A + size\_B];

for (int i = 0; i < size\_A; i++) {

arr[i] = A[i];

}

for (int i = size\_A; i < size\_A + size\_B; i++) {

arr[i] = B[i - size\_A];

}

cout << "Неотсортированный массив, состоящий из множеств А и В:" << endl;

for (int i = 0; i < size\_A + size\_B; i++) {

cout << arr[i] << " ";

}

Merge\_sort(0, size\_A + size\_B - 1, arr, size\_A + size\_B);

cout << endl << "Oтсортированный массив:" << endl;

for (int i = 0; i < size\_A + size\_B; i++) {

cout << arr[i] << " ";

}

delete[]arr;

}

void Merge\_sort(int Left, int Right, int\* arr, int N) { //cортировка слиянием

if (Right == Left) {

return;

}

if (Right - Left == 1) {

if (arr[Right] < arr[Left]) {

std::swap(arr[Right], arr[Left]);

}

return;

}

int middle = (Right + Left) / 2;

Merge\_sort(Left, middle, arr, N);

Merge\_sort(middle + 1, Right, arr, N);

int\* buf = new int[N],

xl = Left,

xr = middle + 1,

cur = 0;

while (Right - Left + 1 != cur) {

if (xl > middle) {

buf[cur++] = arr[xr++];

}

else if (xr > Right) {

buf[cur++] = arr[xl++];

}

else if (arr[xl] > arr[xr]) {

buf[cur++] = arr[xr++];

}

else buf[cur++] = arr[xl++];

}

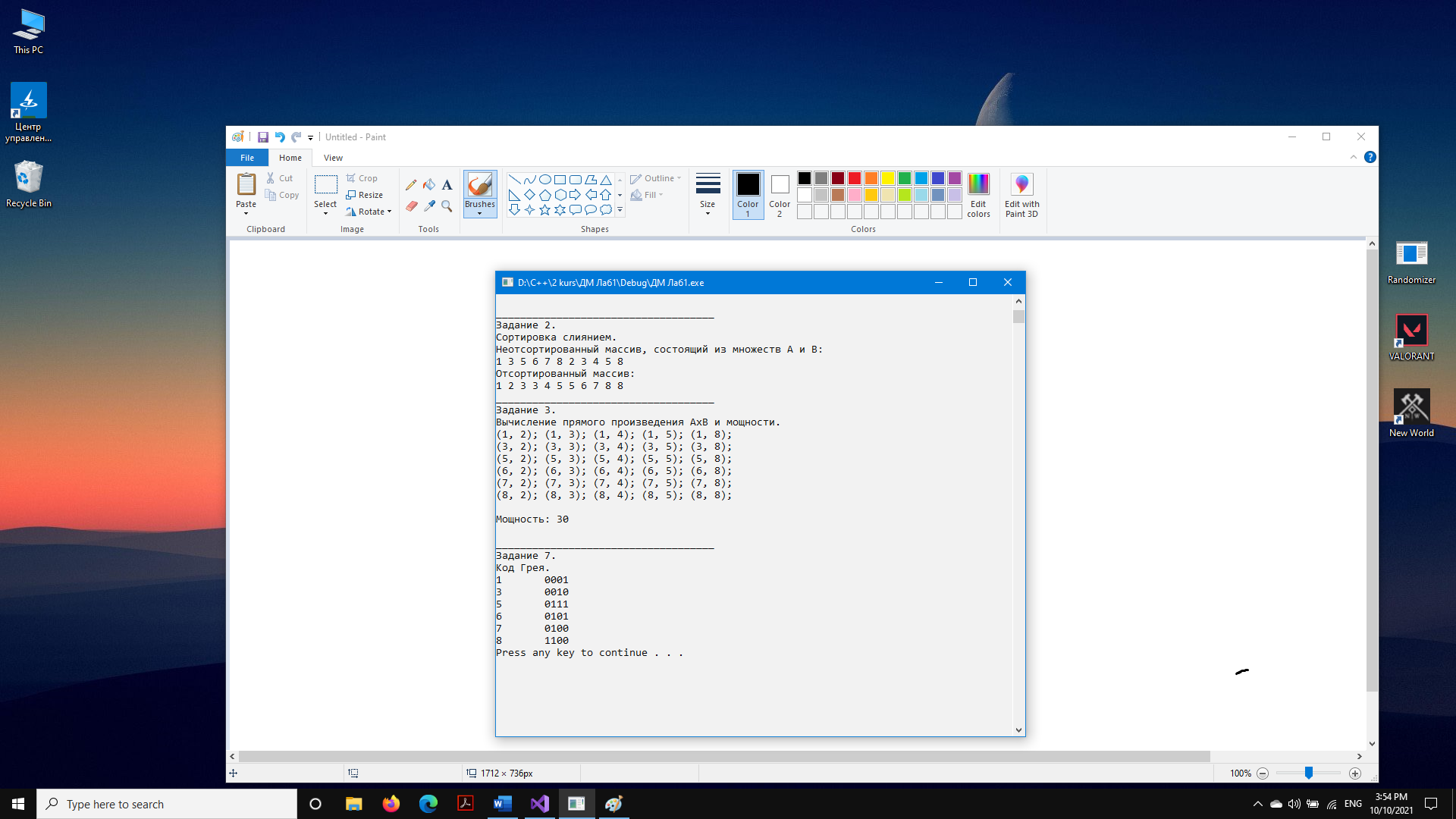
for (int i = 0; i < cur; i++) {

arr[i + Left] = buf[i];

}

delete[] buf;

}



1. Найти прямое произведение  и его мощность.

void t3(int A[], int B[], int size\_A, int size\_B)

{

cout << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "3" << endl;

cout << "Вычисление прямого произведения АхВ и мощности." << endl;

int a = size\_A,

b = size\_B;

int n = a \* b; //мощность по формуле

for (int i = 0; i < a; i++) {

for (int j = 0; j < b; j++) {

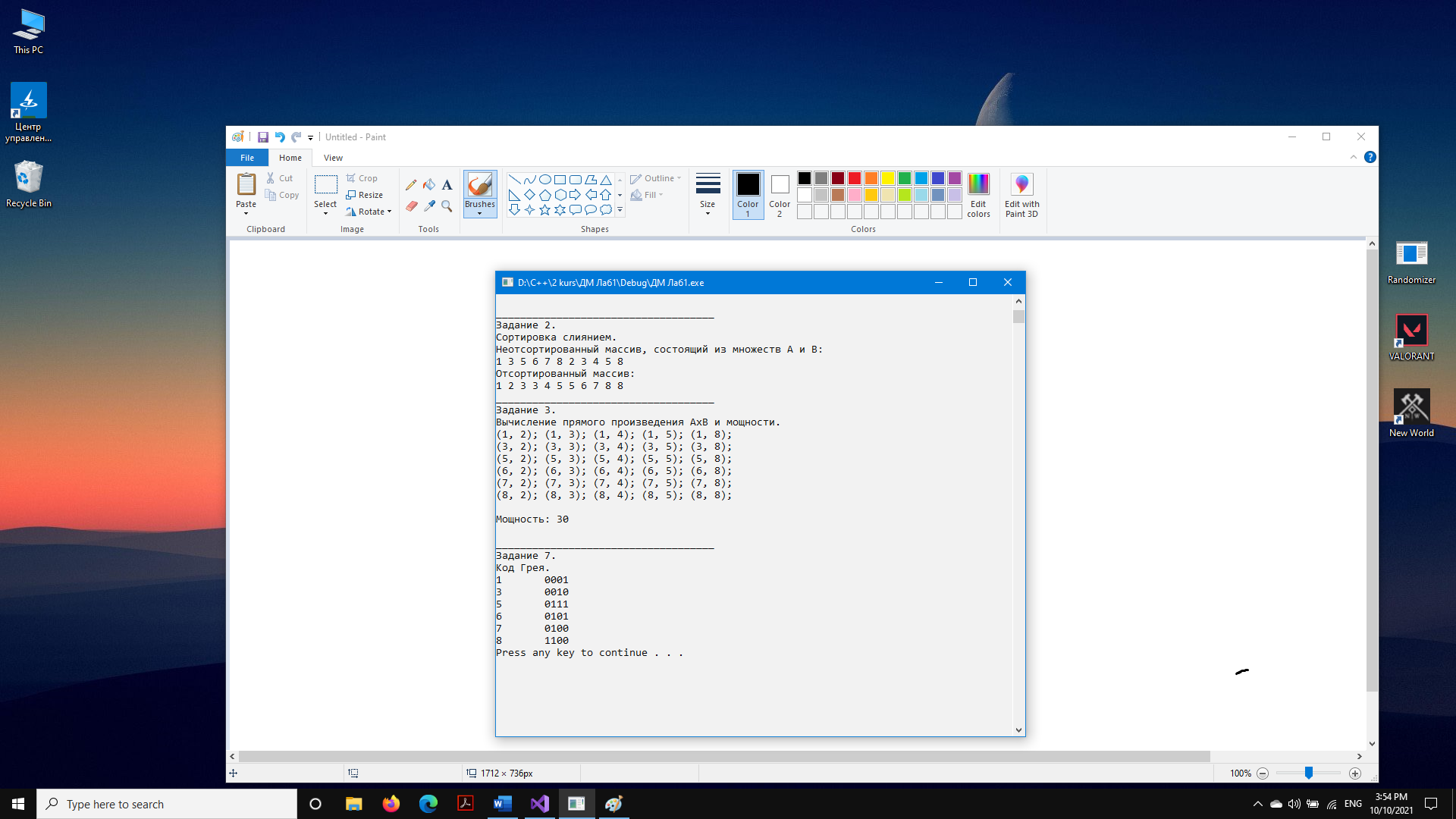
cout << "(" << A[i] << ", " << B[j] << "); ";

}cout << endl;

}

cout << endl << "Мощность: " << n << endl;

}



1. Вычислить программно выражение, используя битовую маску, согласно варианту, проиллюстрировать результат диаграммой *Эйлера-Венна*.

void t4(int A[], int B[], int C[], int U[]) {

cout << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Задание 4." << endl;

cout << "Вычисление выражения согласно варианту: " << endl;

cout << " \_" << endl;

cout << "A U (B n С) = ";

int size\_U = 11,

size\_A = 6,

size\_B = 5,

size\_C = 5,

size = 11,

\* temp = new int[size];

//не B

for (int i = 0; i < size\_U; i++) {

temp[i] = U[i];

}

for (int i = 0; i < size\_B; i++) {

for (int j = 0; j < size; j++) {

if (temp[j] == B[i]) {

for (int k = j; k < size - 1; k++) {

temp[k] = temp[k + 1];

}

j--;

size--;

}

}

}

int\* notB = new int[size\_U];

for (int i = 0; i < size; i++) {

notB[i] = temp[i];

}

//пересечение не В и С

size = 0;

for (int i = 0; i < size\_B; i++) {

for (int j = 0; j < size\_U; j++) {

if (notB[j] == C[i]) {

temp[i] = C[i];

size++;

}

}

}

//объединение А с (не В и С)

list2\* head\_2 = NULL;

int\* intersection\_notB\_C = new int[size];

for (int i = 0; i < size; i++) {

intersection\_notB\_C[i] = temp[i];

create\_node(head\_2, intersection\_notB\_C[i]);

}

list1\* head\_1 = NULL;

for (int i = 0; i < size\_A; i++) {

create\_node(head\_1, A[i]);

}

list1\* q1 = head\_1;

list2\* q2 = head\_2;

while (q1) {

while (q2) {

if (q2->data < q1->data) {

list1\* new\_node = new list1;

new\_node->data = q2->data;

if (q1 == head\_1) {

new\_node->next = head\_1;

head\_1 = new\_node;

}

else {

new\_node->next = q1->next;

q1->next = new\_node;

q1 = q1->next;

}

}

q2 = q2->next;

}

q1 = q1->next;

}

cout << "{";

output(head\_1);

cout << "}" << endl;

delete[] intersection\_notB\_C;

delete[] notB;

delete[] temp;

}

void create\_node(list1\*& head, int data) { //добавление записи в список1

list1\* new\_node = new list1;

new\_node->data = data;

new\_node->next = NULL;

if (head == NULL) {

head = new\_node;

}

else {

list1\* q = head;

while (q->next) {

q = q->next;

}

q->next = new\_node;

}

}

void create\_node(list2\*& head, int data) { //добавление записи в список2

list2\* new\_node = new list2;

new\_node->data = data;

new\_node->next = NULL;

if (head == NULL) {

head = new\_node;

}

else {

list2\* q = head;

while (q->next) {

q = q->next;

}

q->next = new\_node;

}

}

void output(list1\* head) { //вывод списка на экран

for (int i = 0; head; i++) {

cout << head->data;

if (head->next != NULL) {

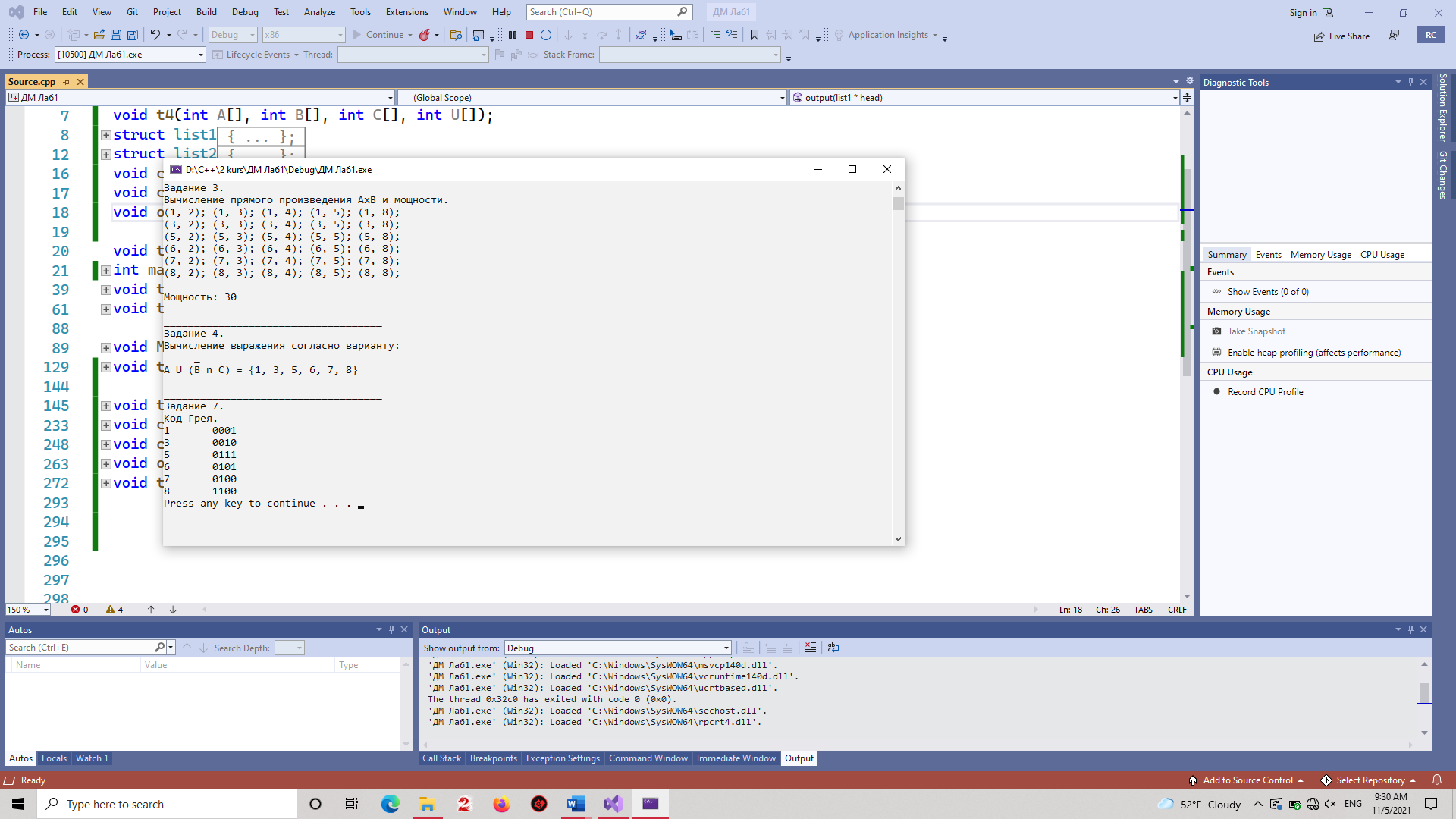
cout << ", ";

}

head = head->next;

}

}



1. Указать примеры покрытий и разбиения множества *A*.

Разбиение множества А:

E1 = {{1, 3, 5, 6, 7, 8}}

E2 = {{1}, {3, 5, 6, 7, 8}}

E3 = {{3}, {1, 5, 6, 7, 8}}

E4 = {{5}, {1, 3, 6, 7, 8}}

E5 = {{6}, {1, 3, 5, 7, 8}}

E6 = {{7}, {1, 3, 5, 6, 8}}

E7 = {{8}, {1, 3, 5, 6, 7}}

E8 = {{1, 3}, {5,6,7,8}}

E9 = {{1,5}, {3,6,7,8}}

E10 = {{1,6}, {3,5,7,8}}

E11 = {{1,7}, {3,5,6,8}}

E12 = {{1,8}, {3,5,6,7}}

E13 = {{1,3,5}, {6,7,8}}

E14 = {{1,5,6}, {3,7,8}}

E15 = {{1,6,7}, {3,5,8}}

E16 = {{1,7,8}, {3,5,6}}

E17 = {{1,6,7,8}, {3,5}}

E18 = {{1,5,7,8}, {3,6}}

E19 = {{1,5,6,8}, {3,7}}

E20 = {{1,5,6,7}, {3,8}}

E21 = {{1,3,7,8}, {5,6}}

E22 = {{1,3,6,8}, {5,7}}

E23 = {{1,3,6,7}, {,5,8}}

E24 = {{1,3,5,8}, {6,7}}

E25 = {{1,3,5,7}, {6,8}}

E26 = {{1,3,5,6}, {7,8}}

E27 = {{1}, {3}, {5}, {6}, {7}, {8}}

Примеры покрытий множества А:

E28 = {{1,3}, {3, 5}, {5,6}, {6,7}, {7,8}}

E29 = {{1,3,5}, {5,6,7,8}}

E30 = {{1,3}, {5,6,7}, {6,8}}

1. \* Реализовать программно *алгоритм раазбиения* множества.
2. Указать для номера вашего варианта (порядок генерации) *бинарный код Грея*.

void t7(int A[], int size\_A)

{

cout << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "7" << endl;

cout << "Код Грея." << endl;

int\* gray\_code = new int[size\_A];

for (int i = 0; i < size\_A; i++) {

cout << A[i] << '\t';

A[i] = A[i] ^ (A[i] >> 1);

for (int j = 3; j >= 0; j--) {

gray\_code[j] = A[i] % 2;

A[i] /= 2;

}

for (int j = 0; j < 4; j++) {

cout << gray\_code[j];

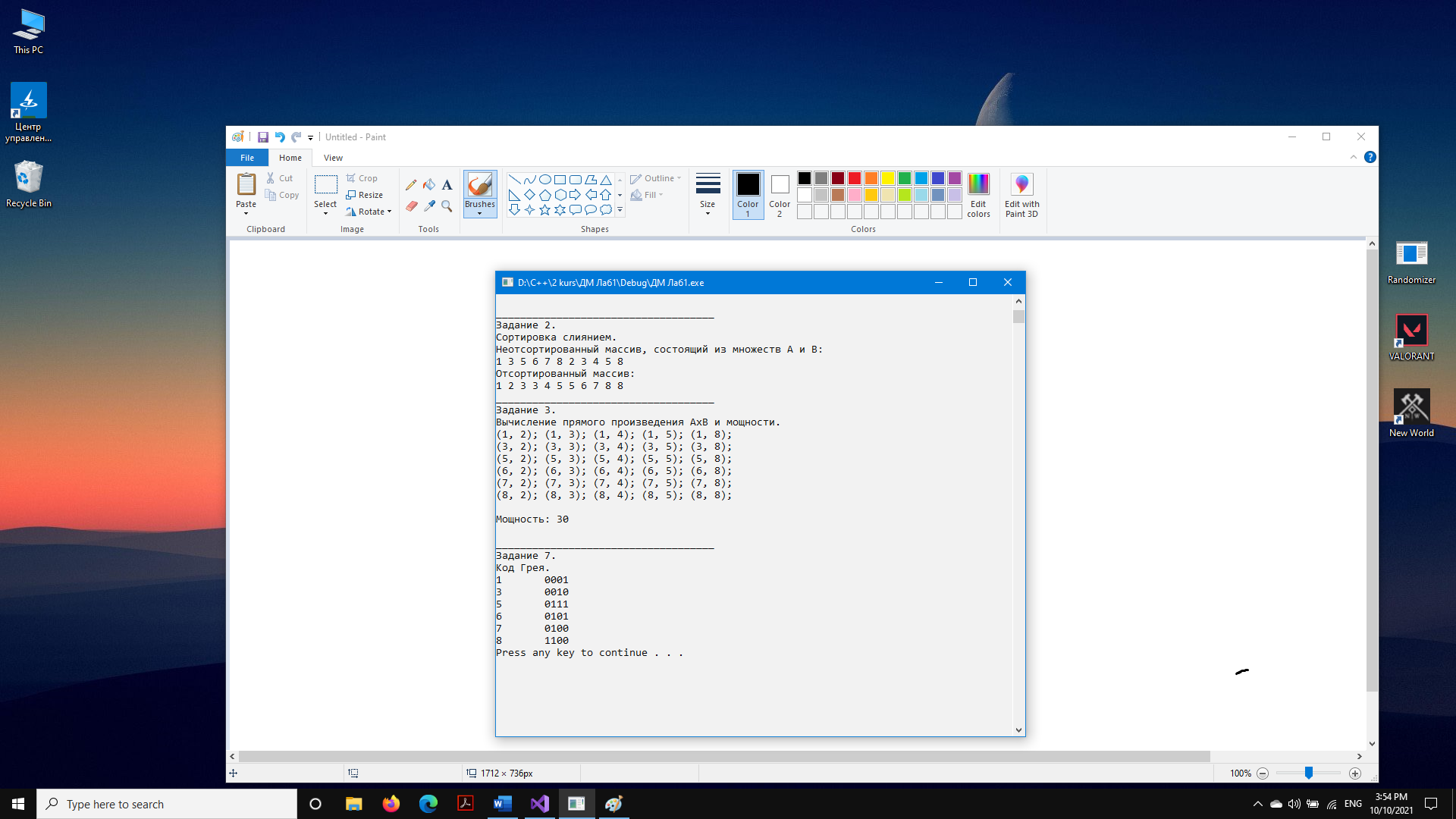
}

cout << endl;

}

delete[] gray\_code;

}



***Задание 2.***

**Вариант 3**

1. В группе из *100* студентов *35* изучают французский язык, *42* - испанский, *43* - немецкий, *17* изучают французский и испанский, *15* - испанский и немецкий, *13* - французский и немецкий, и *20* студентов не изучают ни один из трех языков.

а) Сколько студентов изучают французский или немецкий язык, но не и

изучают испанский?

б) Сколько студентов изучают только один из трех языков?

в) Сколько студентов изучают два из трех языков?

г) Сколько студентов не изучают ни испанский язык, ни французский?

д) Сколько студентов изучают только испанский язык?

Пусть универсум U – множество из 100 студентов, Ф – множество студентов, изучающих французский язык, И – множество

студентов, изучающих испанский язык, Н – множество студентов, изучающих немецкий язык. Предварительно вычислим

два множества. Множество студентов, изучающих хотя бы один из трех перечисленных предметов, равно

|Х ∪ М ∪ Ф| = 100 – 20 = 80.

Тогда множество студентов, изучающих все три предмета, равно

|Х ∩ М ∩ Ф| = |Х ∪ М ∪ Ф| – (|Ф| + |И| + |Н| – |Ф ∩ И| – |И ∩ Н| – |Н ∩ Ф|) = 80 – (35 + 42 + 43 – 17 – 15 – 13) = 80 – 75 = 5.

Множество студентов, изучающих французский или немецкий язык, но не изучающих испанский язык равно

(|Ф| + |Н| – |Ф ∩ Н|) – (|Ф ∩ И| + |И ∩ Н| – |Х ∩ М ∩ Ф|) = (35 + 43 – 13) – (17 + 15 – 5) = 65 – 27 = 38.

Множество студентов, изучающих только один из трех перечисленных языков, равно

(|Ф| – |Ф ∩ И| – |Ф ∩ Н| + |Ф ∩ И ∩ Н|) + (|И| – |И ∩ Ф| – |И ∩ Н| + |Ф ∩ И ∩ Н|) + (|Н| – |Н ∩ Ф| – |Н ∩ И| + |Ф ∩ И ∩ Н|) =

(35 – 17 – 13 + 5) + (42 – 17 – 15 + 5) + (43 – 13 – 15 + 5) = 10 + 15 + 20 = 45.

Множество студентов, изучающих два из трех языков, равно

(|Ф ∩ И| – |Ф ∩ И ∩ Н|) + (|И ∩ Н| – |Ф ∩ И ∩ Н|) + (|Н ∩ Ф| – |Ф ∩ И ∩ Н|) = (17 – 5) + (15 – 5) + (13 – 5) = 12 + 10 + 8 = 30.

Множество студентов, изучающих только испанский язык, равно

|И| – |И ∩ Ф| – |И ∩ Н| + |Ф ∩ И ∩ Н| = 42 – 17 – 15 + 5 = 15.



а) Сколько студентов изучают французский или немецкий язык, но не и изучают испанский?

Ответ: 38.

б) Сколько студентов изучают только один из трех языков?

Ответ: 45.

в) Сколько студентов изучают два из трех языков?

Ответ: 30.

г) Сколько студентов не изучают ни испанский язык, ни французский?

Ответ: 20.

д) Сколько студентов изучают только испанский язык?

Ответ: 15.

***Задание 3***

1. Какова временная сложность алгоритма?

Рабин-Карп усовершенствовал эту концепцию, используя тот факт, что сравнение хэшей двух строк может выполняться за линейное время и намного эффективнее, чем сравнение отдельных символов этих строк для поиска совпадения. Таким образом, обеспечивается наилучшая временная сложность выполнения O (n + m)  
  
2. Какова пространственная (емкостная) сложность алгоритма?  
  
Пространственная сложность - O (m)

3. Что такое коллизии и какие методы позволяют их избежать при данной реализации алгоритма?  
  
Коллизия — совпадение их хешей. В таких случаях необходимо посимвольно проверять совпадение самих подстрок, что занимает достаточно много времени, если данные подстроки имеют большую длину (эту проверку делать не нужно, если ваше приложение допускает ложные срабатывания). При использовании достаточно хороших хеш-функций (смотрите далее) коллизии случаются крайне редко, и в результате среднее время поиска оказывается невелико.

function RabinKarp(string s[1..n], string sub[1..m])

hsub := hash(sub[1..m])

hs := hash(s[1..m])

for i from 1 to (n-m+1)

if hs = hsub

if s[i..i+m-1] = sub

return i

hs := hash(s[i+1..i+m])

return not found

4. Что такое схема Горнера и для чего она используется, есть ли она в данной реализации?  
  
Схе́ма Го́рнера (или правило Горнера, метод Горнера, метод Руффини-Горнера) — алгоритм вычисления значения [многочлена](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%87%D0%BB%D0%B5%D0%BD), записанного в виде суммы мономов (одночленов), при заданном значении переменной. Метод Горнера позволяет найти [корни](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%B5%D0%BD%D1%8C_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) многочлена, а также вычислить производные полинома в заданной точке.x − c {\displaystyle x-c}

5. В каких задачах применяется алгоритм Рабина-Карпа?  
  
Поиск под строки в строке.

Вывод: изучил базовые понятия теории множеств и операции над множествами.