

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных
систем

Лабораторная работа № 1.1

по дисциплине: Дискретная математика
тема: «Операции над множествами»

Выполнил: ст. группы ПВ202

Аладиб язан

Проверил:

Рязанов Юрий Дмитриевич

Бондаренко Татьяна

Владимировна

Лабораторная работа № 1.1

«Операции над множествами»

Цель работы:

изучить и научиться использовать алгебру подмножеств, изучить различные способы представления множеств в памяти ЭВМ, научиться программно реализовывать операции над множествами и выражения в алгебре подмножеств.

Задания:

1. Вычислить значение выражения (см. —Варианты заданий, п. а). Во всех вариантах считать $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Решение изобразить с помощью кругов Эйлера.
2. Записать выражение в алгебре подмножеств, значение которого при заданных множествах A , B и C равно множеству D (см. —Варианты заданий, п. б).
3. Программно реализовать операции над множествами, используя следующие способы представления множества в памяти ЭВМ:
 - а) элементы множества A хранятся в массиве A . Элементы массива A неупорядочены;
 - б) элементы множества A хранятся в массиве A . Элементы массива A упорядочены по возрастанию;
 - в) элементы множества A хранятся в массиве A , элементы которого типа *boolean*. Если $i \in A$, то $A_i = true$, иначе $A_i = false$.
4. Написать программы для вычисления значений выражений (см. —Задания, п.1 и п.2).
5. Используя программы (см. —Задания, п.4), вычислить значения выражений (см. —Задания, п.1 и п.2).

Задание варианта № 14:

- а) $D = C - (A \Delta B) \cup (A \Delta B - C)$
 $A = \{1, 2, 3, 8\}$ $B = \{3, 6, 7\}$ $C = \{2, 3, 4, 5, 7\}$
- б) $A = \{1, 2, 3, 4, 8\}$ $B = \{2, 3, 8\}$ $C = \{3, 4, 5, 6, 7\}$
 $D = \{2, 5, 6, 7, 8\}$

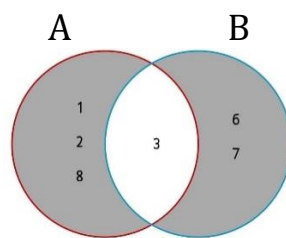
Выполнение работы:

1. Вычислить значение выражения (см. — ”Варианты заданий”, п. а). Во всех вариантах считать $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Решение изобразить с помощью кругов Эйлера.

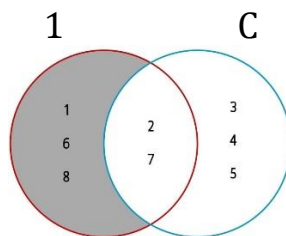
$$a) D = C - (A \Delta B) \cup (A \Delta B - C)$$

$$A = \{1, 2, 3, 8\} \quad B = \{3, 6, 7\} \quad C = \{2, 3, 4, 5, 7\}$$

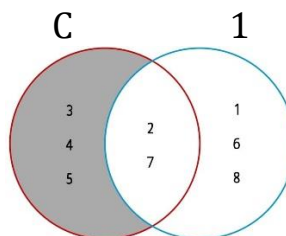
Первое действие: $A \Delta B = \{1, 2, 6, 7, 8\}$



Второе действие: $1 - C = \{1, 6, 8\}$

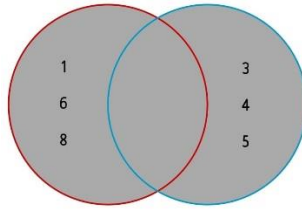


Третье действие: $C - 1 = \{3, 4, 5\}$



Четвёртое действие: $D = 3 \cup 4 = \{1, 3, 4, 5, 6, 8\}$

3 4



2. Записать выражение в алгебре подмножеств, значение которого при заданных множествах A , B и C равно множеству D (см. — “Варианты заданий”, п. б).

$$\begin{aligned} \text{б) } A &= \{1, 2, 3, 4, 8\} \quad B = \{2, 3, 8\} \quad C = \{3, 4, 5, 6, 7\} \\ D &= \{2, 5, 6, 7, 8\} \end{aligned}$$

Первое действие: $C - A = \{5, 6, 7\}$

Второе действие: $B - C = \{2, 8\}$

Третье действие(ответ): $D = (C - A) \cup (B - C) = \{2, 5, 6, 7, 8\}$

3. Программно реализовать операции над множествами, используя следующие способы представления множества в памяти ЭВМ:

а) элементы множества A хранятся в массиве A . Элементы массива A неупорядочены;

б) элементы множества A хранятся в массиве A . Элементы массива A упорядочены по возрастанию;

в) элементы множества A хранятся в массиве A , элементы которого типа *boolean*. Если $i \in A$, то $A_i = \text{true}$, иначе $A_i = \text{false}$.

4. Написать программы для вычисления значений выражений (см. —Задания, п.1 и п.2).

5. Используя программы (см. —Задания, п.4), вычислить значения выражений (см. —Задания, п.1 и п.2).

a)

```
#include <stdio.h>
#include <windows.h>

/* возвращает 1, если в массиве a размера n есть элемент со значением value,
   иначе 0 */
int is_in_set(int *a, int n, int value){
    int i;
    for(i = 0; i < n; ++i)
        if(a[i] == value)
            return 1;
    return 0;
}

/* производит операцию объединения множеств a размера n и b размера m,
   результат записывает в массив res,
   возвращает размер массива res */
int union_sets(int *a, int *b, int n, int m, int *res){
    int i;
    for(i = 0; i < n; ++i)
        res[i] = a[i];
    int c = n;
    for(i = 0; i < m; ++i)
        if(!is_in_set(a, n, b[i])){
            res[c] = b[i];
            ++c;
        }
    return c;
}

/* производит операцию пересечения множеств a размера n и b размера m,
   результат записывает в массив res,
   возвращает размер массива res */
int intersec_sets(int *a, int *b, int n, int m, int *res){
    int c = 0;
    int i;
    for(i = 0; i < n; ++i)
        if(is_in_set(b, m, a[i])){
            res[c] = a[i];
            ++c;
        }
    return c;
}

/* находит дополнение множества a размера n в универсале с границами min и
   max,
   результат записывает в массив res,
```

*возвращает размер массива res */*

```
int comp_set(int *a, int n, int min, int max, int *res){
    int c = 0;
    int i;
    for(i = min; i <= max; ++i)
        if(!is_in_set(a, n, i)){
            res[c] = i;
            ++c;
        }
    return c;
}
```

/ производит операцию разности множеств a размера n и b размера m,
результат записывает в массив res,
возвращает размер массива res */*

```
int diff_sets(int *a, int *b, int n, int m, int *res){
    int c = 0;
    int i;
    for(i = 0; i < n; ++i)
        if(!is_in_set(b, m, a[i])){
            res[c] = a[i];
            ++c;
        }
    return c;
}
```

/ производит операцию симметрической разности
множеств a размера n и b размера m,
результат записывает в массив res,
возвращает размер массива res */*

```
int symm_diff_sets(int *a, int *b, int n, int m, int *res){
    int c = diff_sets(a, b, n, m, res);
    int t = diff_sets(b, a, m, n, res+c);
    return c+t;
}
```

/ возвращает 1, если множество a размера n
является подмножеством множества b размера m,
иначе 0 */*

```
int is_subset(int *a, int *b, int n, int m){
    int i;
    for(i = 0; i < n; ++i){
        if(!is_in_set(b, m, a[i])) return 0;
    }
    return 1;
}
```

/ возвращает 1, если множество a размера n равно множеству b размера m,
иначе 0 */*

```
int is_equal(int *a, int *b, int n, int m){
```

```

    if(is_subset(a, b, n, m) && is_subset(b, a, m, n))
        return 1;
    else
        return 0;
}

void input_arr(int *a, int n){
    int i;
    for(i = 0; i < n; ++i)
        scanf("%i", &a[i]);
}

void output_arr(int *a, int n){
    int i;
    for(i = 0; i < n; ++i)
        printf("%i ", a[i]);
}

int main(){
    SetConsoleOutputCP(CP_UTF8);//Подключение русского языка
    printf("Задание один:\n");
    int n, m, k;
    printf("Введите размер множества A: ");
    scanf("%d", &n);
    int *a = calloc(n, sizeof(int));
    printf("Введите множество A:\n");
    input_arr(a, n);
    printf("Введите размер множества B: ");
    scanf("%d", &m);
    int *b = calloc(m, sizeof(int));
    printf("Введите множество B:\n");
    input_arr(b, m);
    printf("Введите размер множества C: ");
    scanf("%d", &k);
    int *c = calloc(k, sizeof(int));
    printf("Введите множество C:\n");
    input_arr(c, k);

    int *step1, *step2, *step3, *step4;
    step1 = malloc(sizeof(int));
    step2 = malloc(sizeof(int));
    step3 = malloc(sizeof(int));
    step4 = malloc(sizeof(int));
    int size1, size2, size3, size4;

    size1 = symm_diff_sets(a,b,n,m,step1);
    size2 = diff_sets(step1,c,size1,k,step2);
    size3 = diff_sets(c,step1,k,size1,step3);
    size4 = union_sets(step2,step3,size2,size3,step4);
    printf("Искомое множество:\n");

```

```

output_arr(step4, size4);

printf("\n\nЗадание два:\n");
printf("Введите размер множества A: ");
scanf("%d", &n);
printf("Введите множество A:\n");
input_arr(a, n);
printf("Введите размер множества B: ");
scanf("%d", &m);
printf("Введите множество B:\n");
input_arr(b, m);
printf("Введите размер множества C: ");
scanf("%d", &k);
printf("Введите множество C:\n");
input_arr(c, k);
printf("Введите размер множества D: ");
int size_res;
scanf("%d", &size_res);
int d[20];
printf("Введите множество D:\n");
input_arr(d, size_res);

size1 = diff_sets(c,a,k,n,step1);
size2 = diff_sets(b,c,m,k,step2);
size3 = union_sets(step1,step2,size1,size2,step3);
printf("Результат полученного выражения:\n");
output_arr(step3, size3);
printf("\nМножество D:\n");
output_arr(d, size_res);
if(is_equal(step3, d, size3, size_res))
    printf("\nЗначения совпали!");
else
    printf("\nЗначения не совпали!");
return 0;
}

```

b)

```

#include <stdio.h>
#include <malloc.h>
#include <windows.h>

```

```

typedef struct
{
    int* values;
    int n;
}

```



```

    int size;
} sortset;

sortset create_sortset(int size)
{
    sortset res;
    res.n = 0;
    res.size = size;
    res.values = (int*)malloc(size * sizeof(int));
    return res;
}

void delete_sortset(sortset* p_a)
{
    free(p_a->values);
}

void append(sortset* p_a, int value)
{
    if (p_a->n == p_a->size)
    {
        p_a->size *= 2;
        p_a->values = (int*)realloc(p_a->values, p_a->size * sizeof(int));
    }
    (p_a->values)[p_a->n] = value;
    ++(p_a->n);
}

void append_values(sortset* p_a, int n)
{
    for (int i = 0; i < n; ++i)
    {
        int value;
        scanf("%i", &value);
        append(p_a, value);
    }
}

void output_sortset(sortset a)
{
    for (int i = 0; i < a.n; ++i)
        printf("%i ", a.values[i]);
}

char is_in_set(sortset a, int value)
{
    int i = 0;
    while (i < a.n && a.values[i] <= value)
    {

```

```

        if (a.values[i] == value) return 1;
        ++i;
    }
    return 0;
}
char is_equal(sortset a, sortset b)
{
    if (a.n != b.n) return 0;
    for (int i = 0; i < a.n; ++i)
        if (a.values[i] != b.values[i])
            return 0;
    return 1;
}
char is_subset(sortset a, sortset b)
{
    if (a.n > b.n || a.values[0] > b.values[0]) return 0;
    int i = 0;
    int j = 0;
    while (i < a.n && j < b.n)
    {
        if (a.values[i] == b.values[j])
        {
            ++i;
            ++j;
        }
        else if (a.values[i] > b.values[j])
            ++j;
        else return 0;
    }
    return (i == a.n);
}
sortset union_sortsets(sortset a, sortset b)
{
    sortset res = create_sortset(a.size + b.size);
    int i = 0;
    int j = 0;
    while (i < a.n && j < b.n)
    {
        if (a.values[i] == b.values[j])
        {
            append(&res, a.values[i]);
            ++i;
            ++j;
        }
        else if (a.values[i] > b.values[j])

```

```

        {
            append(&res, b.values[j]);
            ++j;
        }
        else
        {
            append(&res, a.values[i]);
            ++i;
        }
    }
    while (i < a.n)
    {
        append(&res, a.values[i]);
        ++i;
    }
    while (j < b.n)
    {
        append(&res, b.values[j]);
        ++j;
    }
    return res;
}

sortset intersec_sortsets(sortset a, sortset b)
{
    sortset res = create_sortset(a.size);
    int i = 0;
    int j = 0;
    while (i < a.n && j < b.n)
    {
        if (a.values[i] == b.values[j])
        {
            append(&res, a.values[i]);
            ++i;
            ++j;
        }
        else if (a.values[i] > b.values[j])
        {
            ++j;
        }
        else
        {
            ++i;
        }
    }
    return res;
}

```

```

}
sortset diff_sortsets(sortset a, sortset b)
{
    sortset res = create_sortset(a.size);
    int i = 0;
    int j = 0;
    while (i < a.n && j < b.n)
    {
        if (a.values[i] == b.values[j])
        {
            ++i;
            ++j;
        }
        else if (a.values[i] > b.values[j])
        {
            ++j;
        }
        else
        {
            append(&res, a.values[i]);
            ++i;
        }
    }
    while (i < a.n)
    {
        append(&res, a.values[i]);
        ++i;
    }
    return res;
}

sortset symdiff_sortsets(sortset a, sortset b)
{
    sortset res = create_sortset(a.size + b.size);
    int i = 0;
    int j = 0;
    while (i < a.n && j < b.n)
    {
        if (a.values[i] == b.values[j])
        {
            ++i;
            ++j;
        }
        else if (a.values[i] > b.values[j])
        {
            append(&res, b.values[j]);

```

```

        ++j;
    }
    else
    {
        append(&res, a.values[i]);
        ++i;
    }
}
while (i < a.n)
{
    append(&res, a.values[i]);
    ++i;
}
while (j < b.n)
{
    append(&res, b.values[j]);
    ++j;
}
return res;
}
sortset compl_sortset(sortset a, int min, int max)
{
    sortset res = create_sortset(max-min+1);
    int u = min;
    int j = 0;
    while (u <= max && j < a.n)
    {
        if (u == a.values[j])
        {
            ++u;
            ++j;
        }
        else if (u > a.values[j])
        {
            ++j;
        }
        else
        {
            append(&res, u);
            ++u;
        }
    }
    while (u <= max)
    {
        append(&res, u);
    }
}

```

```

        ++u;
    }
    return res;
}

int main()
{
    SetConsoleOutputCP(CP_UTF8); //Подключение русского языка
    printf("Задание 1:\n");
    int n, m, k;
    printf("Введите размер множества A: ");
    scanf("%i", &n);
    sortset a = create_sortset(n);
    printf("Введите упорядоченное множество A:\n");
    append_values(&a, a.size);
    printf("Введите размер множества B: ");
    scanf("%i", &m);
    sortset b = create_sortset(m);
    printf("Введите упорядоченное множество B:\n");
    append_values(&b, b.size);
    printf("Введите размер множества C: ");
    scanf("%i", &k);
    sortset c = create_sortset(k);
    printf("Введите упорядоченное множество C:\n");
    append_values(&c, c.size);
    sortset step1 = symdiff_sortsets(a,b);
    sortset step2 = diff_sortsets(step1,c);
    sortset step3 = diff_sortsets(c, step1);
    sortset step4 = union_sortsets(step2,step3);

    printf("Искомое множество:\n");
    output_sortset(step4);
    delete_sortset(&a);
    delete_sortset(&b);
    delete_sortset(&c);
    delete_sortset(&step1);
    delete_sortset(&step2);
    delete_sortset(&step3);
    delete_sortset(&step4);

    printf("\n\nЗадание два:\n");
    printf("Введите размер множества A: ");
    scanf("%i", &n);
    a = create_sortset(n);
    printf("Введите множество A:\n");

```


```

append_values(&a, n);
printf("Введите размер множества B: ");
scanf("%i", &m);
b = create_sortset(m);
printf("Введите множество B:\n");
append_values(&b, m);
printf("Введите размер множества C: ");
scanf("%i", &k);
c = create_sortset(k);
printf("Введите множество C:\n");
append_values(&c, k);
int d_size;
printf("Введите размер множества D: ");
scanf("%i", &d_size);
sortset d = create_sortset(d_size);
printf("Введите множество D:\n");
append_values(&d, d_size);
step1 = diff_sortsets(c,a);
step2 = diff_sortsets(b,c);
step3 = union_sortsets(step1,step2);
printf("Результат полученного выражения:\n");
output_sortset(step3);
printf("\nМножество D:\n");
output_sortset(d);
if (is_equal(step2, d))

    printf("\nЗначения совпали!");
else
    printf("\nЗначения не совпали!");

delete_sortset(&a);
delete_sortset(&b);
delete_sortset(&c);
delete_sortset(&d);
delete_sortset(&step1);
delete_sortset(&step2);
delete_sortset(&step3);
return 0;
}

```



```
Задание один:  
Введите размер множества A:4  
Введите множество A:  
1 2 3 8  
Введите размер множества B:3  
Введите множество B:  
3 6 7  
Введите размер множества C:5  
Введите множество C:  
2 3 4 5 7  
Искомое множество:  
1 3 4 5 6 8  
  
Задание два:  
Введите размер множества A:5  
Введите множество A:  
1 2 3 4 8  
Введите размер множества B:3  
Введите множество B:  
2 3 8  
Введите размер множества C:5  
Введите множество C:  
3 4 5 6 7  
Введите размер множества D:5  
Введите множество D:  
2 5 6 7 8  
Результат полученного выражения:  
2 5 6 7 8  
Множество D:  
2 5 6 7 8  
Значения совпали!  
Process finished with exit code 0
```