МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Операции над множествами с использованием связных списков и алгоритмов слияния

Выполнил студент Чекулаев В.Ю.

Факультет, группа ФКФН, ПО(аб)-81

Проверил Резак Е.В.

Хабаровск – 2020г.

Исходная информация: универсальное множество U, подмножества A, B, C

Множества задаются перечислением элементов. Исходная информация записана в текстовом файле.

Требуется: реализовать операции объединения, пересечения, дополнения, разности для множеств и выполнить вычисления для заданной функции, а также функцию проверки вхождения элемента в множество.

Вариант 1. $\overline{A \cap B \setminus \bar{C}}$

Ручной расчет

Листинг`

```
#include<iostream>
#include<fstream>
#include<string>
#include<stdlib.h>
using namespace std;

struct ELEM{
    int i;
    ELEM *ptr;
};

ELEM* push(int i, ELEM* P){
    ELEM* node = new ELEM;
    node->i = i;
    node->ptr = P;
    return node;
}

ELEM* rev(ELEM* P){
    int n = 0, iter = 0;
    ELEM* result = NULL, *ptr1 = P, *ptr2 = P;

    while(ptr1){
        ptr1 = ptr1->ptr;
        n++;
    }
    int* a = new int[n];
    while(ptr2){
        a[iter] = ptr2->i;
        ptr2 = ptr2->ptr;
        iter++;
    }

    for(int i = 0; i < n; i++){
        result = push(a[i], result);
    }

    return result;
}

void mergeSort(int* a, int l, int r){
    if(l == r) return;
    int mid = (l+r)/2;

    mergeSort(a, l, mid);
    mergeSort(a, mid+1, r);
```

```
    int i = l, j = mid+1;

    int* tmp = new int[r];

    for(int step = 0; step < r-l+1; step++){
        if((j > r) || ((i<=mid) && (a[i] < a[j]))){
            tmp[step] = a[i];
            i++;
        } else{
            tmp[step] = a[j];
            j++;
        }
    }

    for(int step = 0; step < r-l+1; step++){
        a[l+step] = tmp[step];
    }
}

ELEM* createP(string Ps){
    int* a = new int[Ps.length()];
    int c = 0;

    for(int i = 0; i < Ps.length(); i+=2){
        a[c] = (Ps[i]-'0')*10 + (Ps[i+1]-'0');
        c++;
    }

    mergeSort(a, 0, c-1);

    ELEM* tmp = NULL;

    for(int i = c-1; i >= 0; i--){
        tmp = push(a[i], tmp);
    }

    return tmp;
}

bool check(ELEM* P1, ELEM* P2){
    ELEM* a = P1, *b = P2;
    while(a && b){
        if(a->i < b->i){
            return false;
        } else if(a->i > b->i){
            b = b->ptr;
```

```
        } else {
            a = a->ptr;
            b = b->ptr;
        }
    }

    return true;
}

ELEM* combination(ELEM* P1, ELEM* P2){
    ELEM* a = P1, *b = P2, *result = NULL;
    int d;

    while(a && b){
        if(a->i < b->i){
            d = a->i;
            a = a->ptr;
        } else if(a->i > b->i){
            d = b->i;
            b = b->ptr;
        } else{
            d = a->i;
            a = a->ptr;
            b = b->ptr;
        }
        result = push(d, result);
    }

    ELEM* tale = NULL;

    if(a){
        tale = a;
    } else if(b){
        tale = b;
    }

    while(tale){
        result = push(tale->i, result);
        tale = tale->ptr;
    }

    if(!result) return NULL;

    result = rev(result);

    return result;
}
```

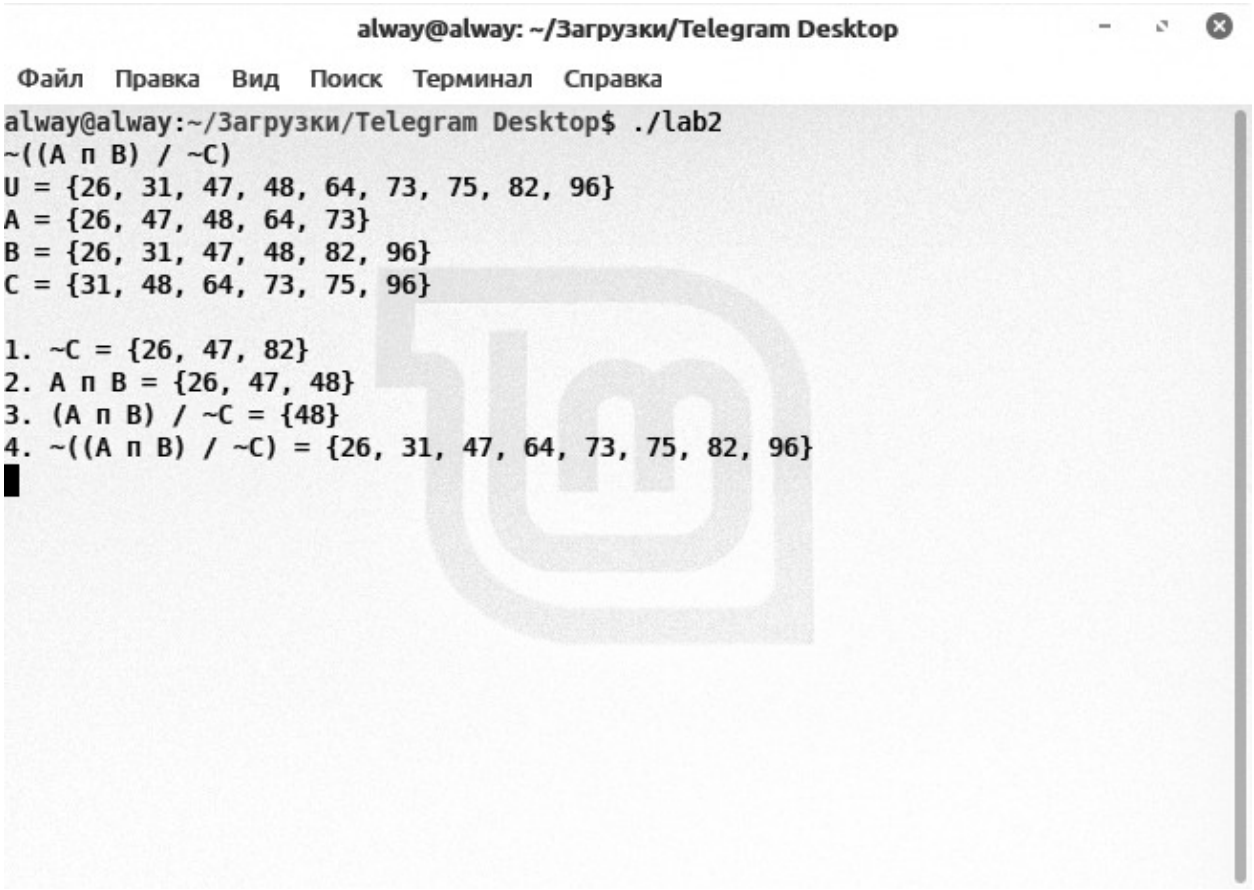
```
ELEM* crossing(ELEM* P1, ELEM* P2){
    ELEM* a = P1, *b = P2, *result = NULL;

    while(a && b){
        if(a->i < b->i){
            a = a->ptr;
        } else if(a->i > b->i){
            b = b->ptr;
        } else{
            result = push(a->i, result);
            a = a->ptr;
            b = b->ptr;
        }
    }

    if(!result) return NULL;

    result = rev(result);

    return result;
}

ELEM* addition(ELEM* P, ELEM* U){
    ELEM* a = P, *u = U, *result = NULL;
    int d;

    while(a && u){
        if(a->i == u->i){
            a = a->ptr;
            u = u->ptr;
        } else if(a->i > u->i){
            result = push(u->i, result);
            u = u->ptr;
        } else if(a->i < u->i){
            a = a->ptr;
        }
    }

    while(u){
        result = push(u->i, result);
        u = u->ptr;
    }

    if(!result) return NULL;

    result = rev(result);
```

```
    return result;
}

int main(){
    string Us, As, Bs, Cs;

    ifstream file ("text1.txt");
    if(!file){
        cout << "File is not open!\n";
        return -1;
    } else{
        file >> Us >> As >> Bs >> Cs;
    }
    file.close();

    ELEM *U, *A, *B, *C;

    U = createP(Us); ELEM* Ubuf = U;
    A = createP(As); ELEM* Abuf = A;
    B = createP(Bs); ELEM* Bbuf = B;
    C = createP(Cs); ELEM* Cbuf = C;

    cout << "~((A ⊓ B) / ~C)\n";

    cout << "U = {";
    while(Ubuf){
        cout << Ubuf->i;
        if(Ubuf->ptr){
            cout << ", ";
        }
        Ubuf = Ubuf->ptr;
    }cout << "}\n";

    cout << "A = {";
    while(Abuf){
        cout << Abuf->i;
        if(Abuf->ptr){
            cout << ", ";
        }
        Abuf = Abuf->ptr;
    }cout << "}\n";

    cout << "B = {";
    while(Bbuf){
        cout << Bbuf->i;
        if(Bbuf->ptr){
```

```
        cout << ", ";
    }
    Bbuf = Bbuf->ptr;
}cout << "}\n";

cout << "C = {";
while(Cbuf){
    cout << Cbuf->i;
    if(Cbuf->ptr){
        cout << ", ";
    }
    Cbuf = Cbuf->ptr;
}cout << "}\n\n";

cout << "1. ~C = {";
ELEM* a1 = addition(C, U);
ELEM* cpy = a1;
while(cpy){
    cout << cpy->i;
    if(cpy->ptr){
        cout << ", ";
    }
    cpy = cpy->ptr;
}cout << "}\n";

cout << "2. A ⊓ B = {";
ELEM* a2 = crossing(A, B);
cpy = a2;
while(cpy){
    cout << cpy->i;
    if(cpy->ptr){
        cout << ", ";
    }
    cpy = cpy->ptr;
}cout << "}\n";

cout << "3. (A ⊓ B) / ~C = {";
ELEM* a31 = addition(a1, U);
ELEM* a32 = crossing(a2, a31);
cpy = a32;
while(cpy){
    cout << cpy->i;
    if(cpy->ptr){
        cout << ", ";
    }
    cpy = cpy->ptr;
}cout << "}\n";
```

```
    cout << "4. ~((A ⊓ B) / ~C) = {";
    ELEM* a4 = addition(a32, U);
    cpy = a4;
    while(cpy){
        cout << cpy->i;
        if(cpy->ptr){
            cout << ", ";
        }
        cpy = cpy->ptr;
    }cout << "}\n";

    getchar();

    return 0;
}
```

Результат работы программы



Вывод

В ходе лабораторной работы были рассмотрены основные операции над множествами с использованием связных списков и алгоритмов слияния. Были проведены ручные расчеты, которые совпали с расчетами программными.