Минобрнауки России

Федеральное государственное автономное образовательное учреждение

высшего образования «Санкт-Петербургский государственный

электротехнический университет «ЛЭТИ»

им В. И. Ульянова (Ленина)»

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

**Зачётная работа № 1**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Множества в памяти ЭВМ»**

Выполнили студенты группы 4315:

Данилова С. В.

Коновалова К. Л.

Принял: старший преподаватель Манирагена В.

Санкт-Петербург

2025

Оглавление

[1. Цель работы 3](#_Toc1)

[2. Задание 3](#_Toc2)

[3. Формула для вычисления пятого множества 3](#_Toc3)

[4. Контрольные тесты 4](#_Toc4)

[5. Временная сложность 5](#_Toc5)

[6. Результат измерения времени обработки для каждого из способов 6](#_Toc6)

[7. Выводы 7](#_Toc7)

[8. Список используемых источников 8](#_Toc8)

[9. Приложение. Текст программы 8](#_Toc9)

[8](#_Toc10)

# 1. Цель работы

Сравнить 4 способа хранения множеств в памяти ЭВМ: set, list, bit array, bit word.

# 2. Задание

Найти множество прописных русских букв, содержащее все буквы множеств A, B, C, которых нет в D.

# 3. Формула для вычисления пятого множества

E = (A ∩ B ∩ C) \ D

# 4. Контрольные тесты

На рис. 1-4 представлены результаты тестов на различных типах множеств.

|  |  |
| --- | --- |
| set<string> |  |
| list<string> |  |
| bit[] –  bit array |  |
| unsigned long long –  bit word |  |

# 5. Временная сложность

Таблица 1. Способы представления и временная сложность обработки

|  |  |  |
| --- | --- | --- |
| Способ представления | Временная сложность | |
| Ожидаемая | Фактическая |
| Множество символов – set | O(n2) | O(n2)  (O(n3) – весь алгоритм) |
| Список – list | O(n2)  (O(n3) – весь алгоритм) |
| Массив битов – bit[] | O(|U|) | O(1) |
| Машинное слово –  unsigned long long | O(1) |

Пояснения:

1. set и list

Для множеств, представленных наборами элементов – set или list, двуместная операция требует проверки всех комбинаций элементов множеств A и B, мощность которых kA и kB , поэтому временная сложность будет O(kA • kB), то есть O(n2).

В программе использовался двойной вложенный цикл для реализации двуместных операций, поэтому фактическая сложность совпадает с ожидаемой.

Но нужно отметить, что в реальной программе также использовался тройной вложенный цикл, который одновременно выполнял две операции – (A ∩ B ∩ C) \ D, а последняя операция выражения – (A ∩ B ∩ C) \ D – выполнялась в двойном вложенном цикле, следующем после тройного. В таком случае получаем сложность реального алгоритма:

O(kA • kB • kC) + O(kmax • kD) = O(kA • kB • kC + kmax • kD) =O(kA • kB • kC) = O(n3)

2) bit array и bit word

Для множеств, представленных отображением на универсум – bit array и bit word, ожидаемое количество шагов двуместной операции равно мощности универсума, то есть O(|U|).

В случае bit array в программе использовался цикл for для выполнения двуместной операции. Цикл проходится по длине массива битов. Его длина – число-константа, поскольку размер универсума изначально известен. Поэтому фактическая временная сложность высчитывается так:

O(|U|) = O(1)

В случае bit word использовались логические операции, поэтому временная сложность буквально уменьшилась до O(1).

# 6. Результат измерения времени обработки для каждого из способов

Таблица 2. Результаты измерения времени обработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мощность множеств | t, с | | | |
| set | list | bit array | bit word |
| 2 | 8e-05 | 9e-05 | 7.7e-05 | 7.5e-05 |
| 4 | 8.9e-05 | 1.1e-04 | 7.6e-05 | 7.7e-05 |
| 6 | 1.1e-04 | 1.2e-04 | 7.8e-05 | 8e-05 |
| 8 | 1.1e-04 | 1.2e-04 | 8.1e-05 | 8e-05 |
| 10 | 1.3e-04 | 1.2e-04 | 8.1e-05 | 7.4e-05 |
| 12 | 1.4e-04 | 1.3e-04 | 8e-05 | 7.7e-05 |
| 14 | 1.7e-04 | 1.5e-04 | 7.6e-05 | 7.6e-05 |
| 16 | 1.7e-04 | 1.7e-04 | 7.6e-05 | 7.8e-05 |
| 18 | 1.8e-04 | 1.9e-04 | 7.8e-05 | 7.9e-05 |
| 20 | 2e-04 | 2.3e-04 | 7.7e-05 | 7.8e-05 |
| 22 | 2.2e-04 | 2.4e-04 | 8e-05 | 8.2e-05 |
| 24 | 2.3e-04 | 2.6e-04 | 8e-05 | 8.1e-05 |
| 26 | 2.4e-04 | 2.6e-04 | 7.6e-05 | 7.9e-05 |
| 28 | 2.7e-04 | 3e-04 | 7.9e-05 | 7.5e-05 |
| 30 | 2.9e-04 | 3.2e-04 | 8.3e-05 | 7.9e-05 |
| 32 | 3.4e-04 | 3.3e-04 | 8.2e-05 | 7.8e-05 |

При представлении множеств в виде set и list заметно, что время обработки множеств с увеличением мощности также увеличивается.

Для bit array и bit word время обработки практически неизменно, поскольку не зависит от размера входа.

# 7. Выводы

# Наиболее быстрым способом представления множеств является машинное слово – bit word, обеспечивающее выполнение операций за O(1).

# Его следует использовать, когда элементы множества можно однозначно сопоставить номерам битов, а мощность универсума не превышает разрядности слова (обычно 64).

# Наиболее медленным оказался список – list, который целесообразно применять только при неизвестной заранее мощности множества и невозможности выделить память под всё множество сразу.

# Таким образом, выбор структуры данных должен основываться на размере универсума, известности мощности множества и требованиях к производительности.

# 8. Список используемых источников

1. Введение в C++. http://www.stolyarov.info/books/pdf/cppintro5.pdf

2. Множества в памяти ЭВМ // Алгоритмы и структуры данных. Лекция от 15.09.2025.

3. Алгоритмы. <http://old.math.nsc.ru/LBRT/k5/OR-MMF/dasgupta_2014.pdf>

4. Студент Д. Рябова. Частное сообщение.

# 9. Приложение. Текст программы

# set

#include <iostream>

#include <string>

#include <chrono>

#include "../headers/set.h"

using namespace std;

string universe = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯЁ";

void init(set<string>& s) {

int size = 22; // -> <= 127 chars in set

int ind;

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

ind = rand()%32+rand()%2;

s.insert(universe.substr(ind\*2, 2));

}

}

void print(set<string>& s) {

for (auto ch : s) cout << ch << " ";

cout << endl;

}

int main() {

auto t1 = std::chrono::high\_resolution\_clock::now();

auto tt1 = clock();

srand(time(0));

setlocale(LC\_ALL, "Russian");

set<string> setA; init(setA);

set<string> setB; init(setB);

set<string> setC; init(setC);

set<string> setD; init(setD);

set<string> setE;

cout << endl;

cout << "A: "; print(setA);

cout << "B: "; print(setB);

cout << "C: "; print(setC);

cout << "D: "; print(setD);

// Находим A ∩ B ∩ C - D

for (auto chA : setA) {

for (auto chB : setB) {

for (auto chC : setC)

if (chA == chB && chA == chC) setE.insert(chA);

}

}

for (auto itE = setE.begin(); itE != setE.end(); ) {

for (auto itD = setD.begin(); itD != setD.end(); itD++) {

if (\*itE == \*itD) {

auto it = itE;

setE.erase(\*it);

break;

}

}

itE++;

}

// Вывод результата

cout << "Множество E = A ∩ B ∩ C - D: "; print(setE);

auto t2 = std::chrono::high\_resolution\_clock::now();

auto tt2 = clock();

cout << "Время выполнения: " << std::chrono::duration\_cast<std::chrono::duration<double, micro>>(t2-t1).count() << " мкс" << endl;

cout << "Счетчик тиков: " << tt2-tt1 << endl;

return 0;

}

# list

#include <iostream>

#include <string>

#include <chrono>

#include "../headers/list.h"

#include "../headers/set.h"

using namespace std;

string universe = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯЁ";

void init(list<string>& l) {

int size = 20;

int ind;

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

ind = rand()%32+rand()%2;

l.push\_back(universe.substr(ind\*2, 2));

}

}

void removeDuplicates(list<string>& l) {

set<string> seen;

auto it = l.begin();

while (it != l.end()) {

if (seen.find(\*it) != nullptr) {

auto itt = it;

l.erase(itt);

} else {

seen.insert(\*it);

}

it++;

}

}

void print(list<string>& l) {

for (auto ch : l) cout << ch << " ";

cout << endl;

}

int main() {

auto t1 = std::chrono::high\_resolution\_clock::now();

auto tt1 = clock();

srand(time(0));

setlocale(LC\_ALL, "Russian");

list<string> listA; init(listA);

list<string> listB; init(listB);

list<string> listC; init(listC);

list<string> listD; init(listD);

list<string> listE;

cout << endl;

cout << "A: "; print(listA); cout << endl;

cout << "B: "; print(listB); cout << endl;

cout << "C: "; print(listC); cout << endl;

cout << "D: "; print(listD); cout << endl;

// Находим A ∩ B ∩ C - D

bool found = 0;

removeDuplicates(listA);

for (auto chA : listA) {

for (auto chB : listB) {

for (auto chC : listC) {

if (chA == chB && chA == chC) {

listE.push\_back(chA); found = 1; break;

}

}

if (found) {found = 0; break;}

}

}

for (auto itE = listE.begin(); itE != listE.end(); ++itE) {

for (auto itD = listD.begin(); itD != listD.end(); ++itD) {

if (\*itE == \*itD) {listE.erase(itE); break;}

}

}

// Вывод результата

cout << "Множество E = A ∩ B ∩ C - D: "; print(listE);

auto t2 = std::chrono::high\_resolution\_clock::now();

auto tt2 = clock();

cout << "Время выполнения: " << std::chrono::duration\_cast<std::chrono::duration<double, micro>>(t2-t1).count() << " мкс" << endl;

cout << "Счетчик тиков: " << tt2-tt1 << endl;

return 0;

}

# bit array

#include <iostream>

#include <stdlib.h>

#include <string>

#include <chrono>

using namespace std;

string universe = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯЁ";

struct bit {

unsigned int bit:1;

};

void init(bit\* b) {

for (int i = 0; i < 33; i++) b[i].bit = rand() % 2;

}

void init\_default(bit\* b) {

for (int i = 0; i < 33; i++) b[i].bit = 0;

}

int f(string s) {

if (s[1]-'0'+160 == -15) return 32; // Ё

return s[1]-'0'+160;

}

string fb(int i) {

return universe.substr(i\*2, 2);

}

void input(bit \*b) {

string line;

getline(cin, line, '\n');

for (int i = 0; i < line.size(); i+=2) b[f(line.substr(i, 2))].bit = 1;

}

void print(bit \*b) {

for (int i = 0; i < 33; i++) {

if (b[i].bit == 1) cout << fb(i) << " ";

}

cout << endl;

}

int main() {

auto t1 = std::chrono::high\_resolution\_clock::now();

auto tt1 = clock();

srand(time(0));

setlocale(LC\_ALL, "Russian");

bit bA[33]; init(bA);

bit bB[33]; init(bB);

bit bC[33]; init(bC);

bit bD[33]; init(bD);

bit bE[33]; init\_default(bE);

cout << endl;

cout << "A: "; print(bA);

cout << "B: "; print(bB);

cout << "C: "; print(bC);

cout << "D: "; print(bD);

// Находим A ∩ B ∩ C - D

for (int i = 0; i < 33; i++) bE[i].bit = bA[i].bit && bB[i].bit && bC[i].bit;

for (int i = 0; i < 33; i++) bE[i].bit = not (bE[i].bit <= bD[i].bit);

// Вывод результата

cout << "Множество E = A ∩ B ∩ C - D: "; print(bE);

auto t2 = std::chrono::high\_resolution\_clock::now();

auto tt2 = clock();

cout << "Время выполнения: " << std::chrono::duration\_cast<std::chrono::duration<double, micro>>(t2-t1).count() << " мкс" << endl;

cout << "Счетчик тиков: " << tt2-tt1 << endl;

return 0;

}

# bit word

#include <iostream>

#include <stdlib.h>

#include <string>

#include <chrono>

using namespace std;

string universe = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯЁ";

unsigned long long init() {

return rand() % 0x2FFFFFFFF;

}

int f(string s) {

if (s[1]-'0'+160 == -15) return 32; // Ё

return s[1]-'0'+160;

}

string fb(int i) {

return universe.substr(i\*2, 2);

}

void input(unsigned long long& w) {

string line;

getline(cin, line, '\n');

for (int i = 0; i < line.size(); i+=2) w |= (1LL << f(line.substr(i, 2)));

}

void show(unsigned long long w) {

for (int i = 0; i < 33; i++) {

std::cout << w%2;

w /= 2;

}

cout << endl;

}

void print(unsigned long long w) {

for (int i = 0; i < 33; i++) {

if (w%2 == 1) cout << fb(i) << " ";

w/=2;

}

cout << endl;

}

int main() {

auto t1 = std::chrono::high\_resolution\_clock::now();

srand(time(0));

setlocale(LC\_ALL, "Russian");

unsigned long long wA = init(); //show(wA);

unsigned long long wB = init(); //show(wB);

unsigned long long wC = init(); //show(wC);

unsigned long long wD = init(); //show(wD);

unsigned long long wE = 0; //show(wE);

cout << endl;

cout << "A: "; print(wA);

cout << "B: "; print(wB);

cout << "C: "; print(wC);

cout << "D: "; print(wD);

// Находим A ∩ B ∩ C - D

wE = wA & wB & wC;

wE = wE & (~wD);

// Вывод результата

cout << "Множество E = A ∩ B ∩ C - D: "; print(wE);

auto t2 = std::chrono::high\_resolution\_clock::now();

cout << "Время выполнения: " << std::chrono::duration\_cast<std::chrono::duration<double, micro>>(t2-t1).count() << " мкс" << endl;

return 0;

}