Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра ВТ

Отчёт

по лабораторной работе № 1

на тему: “Множества”

по дисциплине “Алгоритмы и структуры данных”

Вариант 25

|  |  |  |
| --- | --- | --- |
| Выполнили: | Студенты группы 6306 |  |
|  | Гордиенко М. Е. |  |
|  | Пустовойтова А. А. |  |
| Принял: | Колинько П. Г. |  |

Санкт-Петербург  
2017

Оглавление

Цель 3

Задание 3

Способы задания множества 3

Контрольные примеры 3

Временная сложность 4

Результаты измерения времени обработки 4

Список используемых источников 4

Приложение. Код программы 5

main.cpp 5

mass.h 6

mass.cpp 6

spisok.h 7

spisok.cpp 7

bit\_mass.h 9

bit\_mass.cpp 9

bit.h 10

bit.cpp 10

help\_f.h 11

help\_f.cpp 11

test\_gen.h 12

test\_gen.cpp 12

**Цель**

Исследовать 4 способа хранения множеств в памяти ЭВМ.

**Задание**

Универсум — десятичные цифры.

Вычислить множество, содержащие цифры, имеющиеся в каждом из множеств A, B, C, D.

E = A^B^C^D.

**Способы задания множества**

1. Массив символов
2. Список
3. Массив битов
4. Машинное слово

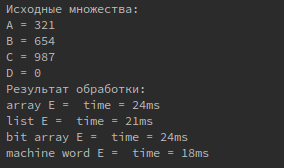
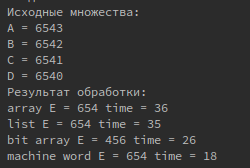
**Контрольные примеры**

Контрольные примеры представлены в таблице 1*.*

Таблица. 1. Контрольные примеры

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Исходные множества | | | | Результат |
| A | B | C | D | E |
| 1 | {6543} | {6542} | {6541} | {6540} | {654} |
| 2 | {123} | {456} | {789} | {0} | { } |

1. Демонстрация работы программы с контрольными примерами 1 и 2 из таблицы 1.



**Временная сложность**

Временная сложность представлена в таблице 2*.*

Таблица. 2. Временная сложность

|  |  |  |
| --- | --- | --- |
| Способ представления | Ожидаемая | Фактическая |
| Массив | O(n^2) | O(n^2) |
| Список | O(n^2) | O(n^2) |
| Машинное слово | O(1) | O(1) |
| Массив битов | O(1) | O(1) |

**Результаты измерения времени обработки**

Результаты измерения времени обработкипредставлены в таблице 3*.*

Таблица. 3. Результаты измерения времени обработки

|  |  |  |  |
| --- | --- | --- | --- |
| Способ представления | Количество миллисекунд | Количество повторов цикла | Зависимость от количества в множестве |
| Массив | ~21-105 | 100000 | есть |
| Список | ~20-129 | есть |
| Машинное слово | ~18 | нет |
| Массив битов | ~26 | нет |

**Вывод:** Представление множества в виде машинного слова самый быстрый из способов обработки данных для рассмотренной задачи, т.к. оно не зависит от количества элементов в множестве.

**Список используемых источников**

* Алгоритмы и структуры данных: методические указания к лабораторным работам, практическим занятиям и курсовому проектированию. Федеральный образовательный стандарт / сост.: П.Г. Колинько. - СПб.: Изд-во СПБГЭТУ "ЛЭТИ", 2017. - 64 с.

**Приложение. Код программы**

**main.cpp**

**#include <iostream>**

#include "mass.h"

#include "test\_gen.h"

#include "help\_f.h"

using namespace std;

int main() {

setlocale(0, " ");

char \*A, \*B, \*C, \*D;

cout << "Выберите пункт меню:\n(1) - Ручной ввод множества.\n(2) - Случайные множества.\n";

int choise;

do {

choise = get\_choise();

switch (choise) {

case 1: {

cout << "Введите множество A: ";

A = hand\_init();

cout << "Введите множество B: ";

B = hand\_init();

cout << "Введите множество C: ";

C = hand\_init();

cout << "Введите множество D: ";

D = hand\_init();

break;

}

case 2: {

A = random\_array();

B = random\_array();

C = random\_array();

D = random\_array();

break;

}

default: {

cout << "Такого пункта нет. Нажмите любую клавишу для повторного выбора.";

cin.get();

system("clear");

break;

}

}

} while (choise != 1 && choise != 2);

test\_program(A, B, C, D);

return 0;

}

**mass.h**

**//**

// Created by matgod on 13.09.17.

//

#ifndef LAB1\_MASS\_H

#define LAB1\_MASS\_H

const int max\_size = 10;

char\* coop(char\* A, char\* B, char\* C, char\* D);

char\* hand\_init();

#endif //LAB1\_MASS\_H

**mass.cpp**

**//**

// Created by matgod on 13.09.17.

//

#include "mass.h"

#include "help\_f.h"

#include <iostream>

char\* coop(char\* A, char\* B, char\* C, char\* D) {

int E\_size = 0;

char\* E = new char [max\_size + 1];

for (int i = 0; A[i] != '\0'; i++) {

for (int j = 0; B[j] != '\0'; j++) {

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

for (int k = 0; C[k] != '\0'; k++) {

if (B[j] == C[k]) {

for (int l = 0; D[l] != '\0'; l++) {

if (C[k] == D[l]) {

E[E\_size++] = D[l];

}

}

}

}

}

}

}

E[E\_size] = '\0';

return E;

}

char\* hand\_init() {

char ch;

char\* mass = new char[max\_size + 1];

int i = 0;

while ((ch = getc(stdin)) != '\n') {

if (!char\_in\_str(ch, mass) && isdigit(ch)) {

mass[i++] = ch;

mass[i] = '\0';

}

}

mass[i] = '\0';

return mass;

}

**spisok.h**

//

// Created by matgod on 13.09.17.

//

#ifndef LAB1\_SPISOK\_H

#define LAB1\_SPISOK\_H

struct list;

list\* to\_list(char\* mass);

list\* coop(list\* A, list\* B, list\* C, list\* D);

void print\_list(list\* head);

void delete\_list(list\* sp);

#endif //LAB1\_SPISOK\_H

**spisok.cpp**

//

// Created by matgod on 13.09.17.

//

#include <cstdlib>

#include "spisok.h"

#include <iostream>

struct list {

char el;

list\* next = NULL;

};

list\* add\_to\_list(list\* head, char ch) {

if (head == NULL) {

head = new list;

head->el = ch;

}

else {

list\* tec = head;

while (tec->next != NULL) {

tec = tec->next;

}

tec->next = new list;

tec->next->el = ch;

}

return head;

}

list\* to\_list(char\* mass) {

int i = 0;

list\* head = NULL;

while (mass[i] != '\0') {

head = add\_to\_list(head, mass[i++]);

}

return head;

}

list\* coop(list\* A, list\* B, list\* C, list\* D) {

list\* E = NULL;

list\* B\_tec;

list\* C\_tec;

list\* D\_tec;

while (A != NULL) {

B\_tec = B;

while (B\_tec != NULL) {

if (A->el == B\_tec->el) {

C\_tec = C;

while (C\_tec != NULL) {

if (B\_tec->el == C\_tec->el) {

D\_tec = D;

while (D\_tec != NULL) {

if (C\_tec->el == D\_tec->el) {

E = add\_to\_list(E, D\_tec->el);

break;

}

D\_tec = D\_tec->next;

}

}

C\_tec = C\_tec->next;

}

}

B\_tec = B\_tec->next;

}

A = A->next;

}

return E;

}

void print\_list(list\* head) {

while (head) {

std::cout << head->el;

head = head->next;

}

}

void delete\_list(list\* sp) {

list\* prev = sp;

while (sp != NULL) {

sp = sp->next;

delete(prev);

prev = sp;

}

}

**bit\_mass.h**

**//**

// Created by matgod on 16.09.17.

//

#ifndef LAB1\_BIT\_MASS\_H

#define LAB1\_BIT\_MASS\_H

bool\* mass\_to\_bit\_mass (char\* mass);

bool\* coop(bool\* b\_m\_A, bool\* b\_m\_B, bool\* b\_m\_C, bool\* b\_m\_D);

void print\_bit\_mass(bool\* bit\_mass);

#endif //LAB1\_BIT\_MASS\_H

**bit\_mass.cpp**

**//**

// Created by matgod on 16.09.17.

//

#include <iostream>

#include "bit\_mass.h"

#include "mass.h"

bool\* mass\_to\_bit\_mass (char\* mass) {

bool\* bit\_mass = new bool[max\_size];

int i;

for (i = 0; i < 10; i++) {

bit\_mass[i] = false;

}

i = 0;

while (mass[i] != '\0') {

bit\_mass[mass[i++] - '0'] = true;

}

return bit\_mass;

}

bool\* coop(bool\* b\_m\_A, bool\* b\_m\_B, bool\* b\_m\_C, bool\* b\_m\_D) {

bool\* b\_m\_E = new bool[max\_size];

for (int i = 0; i < max\_size; i++) {

b\_m\_E[i] = b\_m\_A[i] && b\_m\_B[i] && b\_m\_C[i] && b\_m\_D[i];

}

return b\_m\_E;

}

void print\_bit\_mass(bool\* bit\_mass) {

for (int i = max\_size - 1; i >= 0; i--) {

if (bit\_mass[i]) {

std::cout << i;

}

}

}

**bit.h**

**//**

// Created by matgod on 14.09.17.

//

#ifndef LAB1\_BIT\_H

#define LAB1\_BIT\_H

short mass\_to\_bit(char \*mass);

short coop(short b\_A, short b\_B, short b\_C, short b\_D);

void print\_bit(short);

char\* bit\_to\_mass(short bit);

#endif //LAB1\_BIT\_H

**bit.cpp**

**//**

// Created by matgod on 14.09.17.

//

#include <cmath>

#include <iostream>

#include "bit.h"

#include "mass.h"

short mass\_to\_bit(char \*mass) {

short b\_mass = 0;

for (int i = 0; mass[i] != '\0'; i++) b\_mass += pow(2, mass[i] - '0');

return b\_mass;

}

short coop(short b\_A, short b\_B, short b\_C, short b\_D) {

return b\_A & b\_B & b\_C & b\_D;

}

void print\_bit(short mass) {

short bit\_help = pow(2, 10);

short bit\_num = 10;

while (bit\_help > 1){

if (bit\_help > mass) {

bit\_help /=2;

bit\_num--;

}

else {

std::cout << bit\_num;

mass -= bit\_help;

}

}

if (mass == 1) std::cout << bit\_num;

}

char\* bit\_to\_mass(short bit) {

char\* mass = new char[max\_size + 1];

short bit\_help = (short)pow(2, max\_size);

short bit\_num = 10;

short i = 0;

while (bit\_help > 1){

if (bit\_help > bit) {

bit\_help /=2;

bit\_num--;

}

else {

mass[i++] = (char)(bit\_num + '0');

bit -= bit\_help;

}

}

if (bit == 1) {

mass[i++] = '0';

}

mass[i] = '\0';

return mass;

}

**help\_f.h**

**//**

// Created by matgod on 14.09.17.

//

#ifndef LAB1\_HELP\_F\_H

#define LAB1\_HELP\_F\_H

bool char\_in\_str(char ch, char\* str);

int get\_choise();

char\* random\_array();

#endif //LAB1\_HELP\_F\_H

**help\_f.cpp**

//

// Created by matgod on 14.09.17.

//

#include <iostream>

#include "help\_f.h"

#include "bit.h"

bool char\_in\_str(char ch, char\* str) {

int i = 0;

while (str[i] != '\0') {

if (ch == str[i++]) return true;

}

return false;

}

int get\_choise() {

char ch;

int num = 0;

while ((ch = getc(stdin)) != '\n') {

if (isdigit(ch)) {

num = ch - '0';

}

}

return num;

}

char\* random\_array() {

srand(clock());

return bit\_to\_mass(rand() % 1024);

}

**test\_gen.h**

**//**

// Created by matgod on 19.09.17.

//

#ifndef LAB1\_TEST\_GEN\_H

#define LAB1\_TEST\_GEN\_H

void test\_program(char\* b\_A, char\* b\_B, char\* b\_C, char\* b\_D);

#endif //LAB1\_TEST\_GEN\_H

**test\_gen.cpp**

**//**

// Created by matgod on 19.09.17.

//

#include <cmath>

#include <iostream>

#include "test\_gen.h"

#include "bit.h"

#include "bit\_mass.h"

#include "spisok.h"

#include "mass.h"

using namespace std;

void test\_program(char\* A, char\* B, char\* C, char\* D) {

char \*E;

list \*A\_list = to\_list(A),

\*B\_list = to\_list(B),

\*C\_list = to\_list(C),

\*D\_list = to\_list(D),

\*E\_list;

short b\_A = mass\_to\_bit(A),

b\_B = mass\_to\_bit(B),

b\_C = mass\_to\_bit(C),

b\_D = mass\_to\_bit(D),

b\_E;

bool \*b\_m\_A = mass\_to\_bit\_mass(A),

\*b\_m\_B = mass\_to\_bit\_mass(B),

\*b\_m\_C = mass\_to\_bit\_mass(C),

\*b\_m\_D = mass\_to\_bit\_mass(D),

\*b\_m\_E;

unsigned long start, stop, array\_time = 0, list\_time = 0, bit\_mass\_time = 0, machine\_word\_time = 0,

count = 100000, tic\_in\_ms = CLOCKS\_PER\_SEC / 1000;

for (unsigned long i = 0; i < count; i++) {

start = clock();

E = coop(A, B, C, D);

stop = clock();

array\_time += (stop - start);

start = clock();

E\_list = coop(A\_list, B\_list, C\_list, D\_list);

stop = clock();

list\_time += (stop - start);

start = clock();

b\_m\_E = coop(b\_m\_A, b\_m\_B, b\_m\_C, b\_m\_D);

stop = clock();

bit\_mass\_time += (stop - start);

start = clock();

b\_E = coop(b\_A, b\_B, b\_C, b\_D);

stop = clock();

machine\_word\_time += (stop - start);

if (i < count - 1) {

delete[] E;

E = nullptr;

delete\_list(E\_list);

delete[] b\_m\_E;

b\_m\_E = nullptr;

}

}

cout << "Исходные множества:\nA = ";

print\_bit(b\_A);

cout << "\nB = ";

print\_bit(b\_B);

cout << "\nC = ";

print\_bit(b\_C);

cout << "\nD = ";

print\_bit(b\_D);

cout << "\nРезультат обработки:\narray E = " << E << " time = " << array\_time / tic\_in\_ms << "ms\n";

cout << "list E = ";

print\_list(E\_list);

cout << " time = " << list\_time / tic\_in\_ms << "ms\n";

cout << "bit array E = ";

print\_bit\_mass(b\_m\_E);

cout << " time = " << bit\_mass\_time / tic\_in\_ms << "ms\n";

cout << "machine word E = ";

print\_bit(b\_E);

cout << " time = " << machine\_word\_time / tic\_in\_ms << "ms\n";

delete[] A;

A = nullptr;

delete[] B;

B = nullptr;

delete[] C;

C = nullptr;

delete[] D;

D = nullptr;

delete[] E;

E = nullptr;

delete\_list(A\_list);

delete\_list(B\_list);

delete\_list(C\_list);

delete\_list(D\_list);

delete\_list(E\_list);

delete[] b\_m\_A;

b\_m\_A = nullptr;

delete[] b\_m\_B;

b\_m\_B = nullptr;

delete[] b\_m\_C;

b\_m\_C = nullptr;

delete[] b\_m\_D;

b\_m\_D = nullptr;

delete[] b\_m\_E;

b\_m\_E = nullptr;

}