Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа**

**по курсу «ОC»**

**Тема:**

**Многопоточное программирование.**

|  |  |
| --- | --- |
| Студент: | Суворова С. А. |
| Группа: | М80-206Б-18 |
| Преподаватель: | Соколов А.А. |
| Вариант: | 23 |
| Оценка: |  |
| Дата: |  |

Москва

2019

1.**Код на C:**

vector.h:

#include <stdio.h>

#include <stdbool.h>

#ifndef \_VECTOR\_

#define \_VECTOR\_

typedef struct {

int \*body;

int size;

} vector;

vector \*v\_create();

void v\_destroy(vector \*v);

void v\_set(vector \*v, int i, int val);

int v\_get(vector \*v,int i);

bool v\_set\_size(vector \*v,int size);

int v\_get\_size(vector \*v);

bool v\_push(vector \*v,int val);

#endif

vector.c:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include "vector.h"

vector \*v\_create() {

vector \*v = (vector \*) malloc(sizeof(vector));

v->size = 0;

v->body = NULL;

return v;

}

void v\_destroy(vector \*v) {

free(v->body);

free(v);

}

int v\_get\_size(vector \*v) {

return v->size;

}

int v\_get(vector \*v, int i) {

return v->body[i];

}

bool v\_set\_size(vector \*v, int size) {

int \*bodyre = (int \*) realloc(v->body, size \* sizeof(int));

if (bodyre == NULL && size != 0) {

return false;

}

if(bodyre == NULL){

return true;

}

for (int i = v->size; i<size; i++) {

bodyre[i] = 0;

}

v->body=bodyre;

v->size=size;

return true;

}

void v\_set(vector \*v, int i, int val) {

v->body[i] = val;

}

bool v\_push(vector \*v, int val) {

int new\_size=v->size + 1;

if(!v\_set\_size(v,new\_size)){

return false;

}

v\_set(v,new\_size - 1,val);

return true;

}

main.c:

[#include](https://vk.com/im?sel=157419337&st=%23include) <stdio.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) "vector.h"  
[#include](https://vk.com/im?sel=157419337&st=%23include) <stdlib.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <pthread.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <string.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <unistd.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <math.h>  
  
bool is\_space(char a) {  
if ((a == ' ') || (a == '\n') || (a == '\t') || (a == '\0')) {  
return true;  
} else {  
return false;  
}  
}  
  
bool is\_power\_0f\_two(int n) {  
bool flag;  
if ((n & (n - 1)) == 0) {  
flag = true;  
} else {  
flag = false;  
}  
return flag;  
}  
  
int main() {  
char a[1];  
vector \*v = v\_create();  
ssize\_t y;  
int b;  
bool flag = false;  
int c[10];  
int r = 0;  
read(0, a, 1);  
while (is\_space(a[0])) {  
a[0] = 0;  
y = read(0, a, 1);  
if (y <= 0) {  
break;1010010  
}  
}  
while (!is\_space(a[0])) {  
b = a[0] - '0';  
if ((b > 1) || (b < 0)) {  
flag = true;  
}  
v\_push(v, b);  
a[0] = 0;  
y = read(0, a, 1);  
if (y <= 0) {  
break;  
}  
}  
if (flag == true) {  
write(1, "Неверный ввод\n", 27);  
} else {  
write(1, "Введите колличество контрольных разрядов:\n", 80);  
while (is\_space(a[0])) {  
a[0] = 0;  
y = read(0, a, 1);  
if (y <= 0) {  
break;  
}  
}  
int i = 0;  
while (!is\_space(a[0])) {  
c[i] = a[0] - '0';  
a[0] = 0;  
y = read(0, a, 1);  
if (y <= 0) {  
break;  
}  
++i;  
}  
c[i] = '\0';  
int g = 1;  
int j = 0;  
while (c[j] != '\0') {  
g = g \* 10;  
++j;  
}  
g = g / 10;  
j = 0;  
while (g > 0) {  
r = r + c[j] \* g;  
g = g / 10;  
++j;  
}  
int n, m, k;  
k = r;  
n = 1;  
while (k > 0) {  
n = n \* 2;  
k = k - 1;  
}  
n = n - 1;  
m = 1;  
k = r;  
while (k > 0) {  
m = m \* 2;  
k = k - 1;  
}  
m = m - r - 1;  
int d[m][r];  
int l[m][m];  
int p = 0;  
for (int i1 = 0; i1 < m; ++i1) {  
for (int k1 = 0; k1 < m; ++k1) {  
if (k1 == p) {  
l[i1][k1] = 1;  
} else {  
l[i1][k1] = 0;  
}  
}  
++p;  
}  
int x;  
int q = 0;  
int w = r - 1;  
for (int l1 = n; l1 >= 1; —l1) {  
if (!is\_power\_0f\_two(l1)) {  
x = l1;  
for (int i1 = 0; i1 < r; ++i1) {  
d[q][w] = x % 2;  
x = x / 2;  
—w;  
}  
++q;  
w = r - 1;  
}  
}  
int res[m][n];  
for (int m1 = 0; m1 < m; ++m1) {  
for (int i1 = 0; i1 < n; ++i1) {  
if (i1 < m) {  
res[m1][i1] = l[m1][i1];  
} else {  
res[m1][i1] = d[m1][i1 - m];  
}  
}  
}  
vector \*cod\_word = v\_create();  
int a1;  
int b1;  
int size\_of\_v=v\_get\_size(v);  
int m1=0;  
bool flag=false;  
while(m1<size\_of\_v) {  
for (int n1 = 0; n1 < n; ++n1) {  
a1 = 0;  
for (int i1 = 0; i1 < m; ++i1) {  
b1 = v\_get(v, i1+m1);  
a1 = a1 + b1 \* res[i1][n1];  
}  
if ((a1 % 2) != 0) {  
v\_push(cod\_word, 1);  
} else {  
v\_push(cod\_word, 0);  
}  
}  
m1=m1+m;  
if(size\_of\_v-m1<m){  
if(size\_of\_v-m1!=0){  
flag=true;  
}  
break;  
}  
}  
if(flag==true){  
write(1, "Нельзя закодировать\n", 40);  
}else {  
char charo[1];  
int o;  
for (int l = 0; l < cod\_word->size; ++l) {  
o=v\_get(cod\_word,l);  
charo[0]=(char)o;  
charo[0]=charo[0]+48;  
write(1,charo, 1);  
}  
write(1,"\n", sizeof("\n"));  
}  
}  
return 0;  
}

main1.c:

[#include](https://vk.com/im?sel=157419337&st=%23include) <stdio.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) "vector.h"  
[#include](https://vk.com/im?sel=157419337&st=%23include) <stdlib.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <pthread.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <string.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <unistd.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <math.h>  
  
pthread\_mutex\_t mutex=PTHREAD\_MUTEX\_INITIALIZER;  
int counter=0;  
  
struct args1 {  
vector \*v;  
int start;  
int n;  
int r;  
int \*\*h\_matr;  
int i1;  
int j1;  
int \*\*res;  
int j2;  
};  
  
void \*matr\_multiply(void \*args) {  
pthread\_mutex\_lock(&mutex);  
struct args1 [\*arg](https://vk.com/id209357492) = (struct args1 \*) args;  
int res = 0;  
for (int i = 0; i < arg->n; ++i) {  
res = res + arg->v->body[i+arg->start] \* arg->h\_matr[i][arg->j1];  
}  
res = res % 2;  
arg->res[0][arg->j2] = res;  
++arg->j1;  
++arg->j2;  
++counter;  
pthread\_mutex\_unlock(&mutex);  
return EXIT\_SUCCESS;  
}  
  
bool is\_space(char a) {  
if ((a == ' ') || (a == '\n') || (a == '\t') || (a == '\0')) {  
return true;  
} else {  
return false;  
}  
}  
  
bool is\_power\_0f\_two(int n) {  
bool flag;  
if ((n & (n - 1)) == 0) {  
flag = true;  
} else {  
flag = false;  
}  
return flag;  
}  
  
int what\_string(int n,int r,int h\_matr[n][r], int res[1][r]){  
bool flag=false;  
int g=0;  
for (int i = 0; i < n; ++i) {  
for (int j = 0; j < r; ++j) {  
if(h\_matr[i][j]!=res[0][j]){  
break;  
}else{  
if(j==r-1){  
flag=true;  
}  
}  
}  
if(flag==true){  
break;  
}  
++g;  
}  
if(flag==true){  
return g;  
}else{  
return -1;  
}  
}  
  
int main() {  
char a[1];  
vector \*v = v\_create();  
ssize\_t y;  
int b;  
bool flag = false;  
int c[10];  
int r = 0;  
read(0, a, 1);  
while (is\_space(a[0])) {  
a[0] = 0;  
y = read(0, a, 1);  
if (y <= 0) {  
break;  
}  
}  
while (!is\_space(a[0])) {  
b = a[0] - '0';  
if ((b > 1) || (b < 0)) {  
flag = true;  
}  
v\_push(v, b);  
a[0] = 0;  
y = read(0, a, 1);  
if (y <= 0) {  
break;  
}  
}  
if (flag == true) {  
write(1, "Неверный ввод\n", 27);  
} else {  
write(1, "Введите колличество контрольных разрядов:\n", 80);  
while (is\_space(a[0])) {  
a[0] = 0;  
y = read(0, a, 1);  
if (y <= 0) {  
break;  
}  
}  
int i = 0;  
while (!is\_space(a[0])) {  
c[i] = a[0] - '0';  
a[0] = 0;  
y = read(0, a, 1);  
if (y <= 0) {  
break;  
}  
++i;  
}  
c[i] = '\0';  
int g = 1;  
int j = 0;  
while (c[j] != '\0') {  
g = g \* 10;  
++j;  
}  
g = g / 10;  
j = 0;  
while (g > 0) {  
r = r + c[j] \* g;  
g = g / 10;  
++j;  
}  
int n, m, k;  
k = r;  
n = 1;  
while (k > 0) {  
n = n \* 2;  
k = k - 1;  
}  
n = n - 1;  
m = 1;  
k = r;  
while (k > 0) {  
m = m \* 2;  
k = k - 1;  
}  
m = m - r - 1;  
int l[r][r];  
int p = 0;  
for (int i1 = 0; i1 < r; ++i1) {  
for (int k1 = 0; k1 < r; ++k1) {  
if (k1 == p) {  
l[i1][k1] = 1;  
} else {  
l[i1][k1] = 0;  
}  
}  
++p;  
}  
int d[m][r];  
int x;  
int q = n;  
for (int i1 = 0; i1 < m; ++i1) {  
x = q;  
for (int k1 = r - 1; k1 >= 0; —k1) {  
d[i1][k1] = x % 2;  
x = x / 2;  
}  
q = q - 1;  
if (is\_power\_0f\_two(q)) {  
q = q - 1;  
}  
}  
int h\_matr[n][r];  
for (int m1 = 0; m1 < n; ++m1) {  
for (int i1 = 0; i1 < r; ++i1) {  
if (m1 < m) {  
h\_matr[m1][i1] = d[m1][i1];  
} else {  
h\_matr[m1][i1] = l[m1 - m][i1];  
}  
}  
}  
int n1 = 0;  
bool flag = false;  
struct args1 [\*arg](https://vk.com/id209357492) = malloc(sizeof(struct args1));  
int res[1][r];  
vector \*decod\_word = v\_create();  
arg->res=(int \*\*)malloc(sizeof(int \*));  
for (int j2 = 0; j2 < 1; ++j2) {  
arg->res[j2] = (int \*) malloc(sizeof(int) \* r);  
}  
arg->h\_matr=(int \*\*)malloc(sizeof(int \*)\*n);  
for (int k2 = 0; k2 < n; ++k2) {  
arg->h\_matr[k2] = (int \*) malloc(sizeof(int) \* r);  
}  
for (int i1 = 0; i1 < n; ++i1) {  
for (int k1 = 0; k1 < r; ++k1) {  
arg->h\_matr[i1][k1]=h\_matr[i1][k1];  
}  
}  
pthread\_mutex\_init(&mutex,NULL);  
while (n1 < v->size) {  
arg->start = n1;  
arg->i1 = 0;  
arg->j1 = 0;  
arg->n = n;  
arg->r = r;  
arg->v = v;  
arg->j2 = 0;  
for (int l = 0; l < r; ++l) {  
pthread\_t thread;  
int status;  
status = pthread\_create(&thread, NULL, matr\_multiply, arg);  
if (status != 0) {  
exit(EXIT\_FAILURE);  
}  
pthread\_join(thread,NULL);  
}  
for (int l1 = 0; l1 < 1; ++l1) {  
for (int i1 = 0; i1 < r; ++i1) {  
res[l1][i1]=arg->res[l1][i1];  
}  
}  
int results = 0;  
for (int i1 = 0; i1 < r; ++i1) {  
results = results + pow(2, r-i1-1) \* res[0][i1];  
}  
if (results != 0) {  
int f=what\_string(n,r,h\_matr,res);  
if(is\_power\_0f\_two(results)){  
v->body[n1 + f] = (v->body[n1 + f] + 1) % 2;  
}else {

v->body[n1 + f] = (v->body[n1 + f] + 1) % 2;  
}  
}  
for (int k1 = 0; k1 < m; ++k1) {  
v\_push(decod\_word, v->body[n1 + k1]);  
}  
n1 = n1 + n;  
if (v->size - n1 < n) {  
if (v->size - n1 != 0) {  
flag = true;  
}  
free(arg);  
break;  
}  
}  
if (flag == true) {  
write(1, "Нельзя декодировать\n", 40);  
}else {  
pthread\_mutex\_destroy(&mutex);  
char charo[1];  
int o;  
for (int l = 0; l < decod\_word->size; ++l) {  
o=v\_get(decod\_word,l);  
charo[0]=(char)o;  
charo[0]=charo[0]+48;  
write(1,charo, 1);  
}  
write(1,"\n", sizeof("\n"));  
write(1,&counter, 8);  
}  
}  
return 0;  
}

2. . **Ссылка на репозиторий в GitHub:**

https://github.com/Suvorova-Sofya/os\_lab\_3

3.**Набор тестов:**

test1:

1010  
Введите количество контрольных разрядов:  
3  
1010010

test2:

1200  
Неверный ввод

test3:

1010010  
Введите количество контрольных разрядов:  
3  
1010

4.**Результаты работы программы на тестах:**

test1:

1010  
Введите количество контрольных разрядов:  
3  
1010010

test2:

1200  
Неверный ввод

test3:

1010010  
Введите количество контрольных разрядов:  
3  
1010

5**. Объяснение работы программы:**

Пользователь вводит последовательность 0 и 1, которую хочет закодировать или декодировать используя код Хемминга, и количество контрольных разрядов. Далее программа вычисляет длину кодируемого или кодового слова исходя из количества контрольных разрядов, и если длинна введенного слова не делится без остатка на вычисленную длину, то выдается ошибка. Далее используя все необходимые вычисления получается кодовое или декодированное слово. Многопоточность происходит при перемножении матриц во второй программе(main1.c).

6.**Вывод:**

В данной программе показывается, как можно использовать потоки для одновременного исполнения срезу нескольких вычислений. Таким образом время выполнения программы становится меньше, а производительность не уменьшается.