МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Инженерная школа информационных технологий и робототехники Направление: Информатика и вычислительная техника

Отделение информационных технологий

Отчет по лабораторной работе №3 по дисциплине

«Организация ЭВМ»

«ПРОГРАММНОЕ ВЗАИМОДЕЙСТВИЕ С ПАМЯТЬЮ ЭВМ»

Вариант 28

Выполнил: Студент группы 8В01

И.П. Тюрин

Проверил: Инженер ОИТ

##### Ю.В. Котова

Томск 2022

**Цель работы**

Получить знания о принципах работы с оперативной и кэшпамятью ЭВМ на примере программного взаимодействия с помощью языка C++.

**Задание**

1. Создать 64-битное приложение на С++. Определить экспериментально для своей системы (или для системы аудиторного ПК) максимально возможны размер памяти, динамически выделяемой под массив.
2. Воспроизвести программно ситуацию, при которой выход за границы массива искажает значения других переменных или массива, но не приводит к ошибке доступа к памяти. Создать ситуацию, при которой операционная система выдает ошибку доступа к памяти, используя выход за границы массива.
3. Воспроизвести ситуации из п. 2, используя указатели в C++.
4. Создать и продемонстрировать с помощью C++ простую утечку памяти. Привести способ её устранения.
5. Провести эксперименты с кэш-памятью (№ (example) 3 по ссылке http://igoro.com/archive/gallery-of-processor-cache-effects/) своего процессора (или процессора аудиторного ПК).
6. Для эксперимента № 3 п.5. необходимо запустить программу в цикле с изменяющимся размером массива от 1 Кб до 128 Мб c шагом, кратным степени 2 КБ, начиная с 1 КБ (1,2,4,..,65536,131072 КБ). Для каждого размера массива посчитать среднее время работы. По результатам построить график в Excel (либо в любом другом инструменте или написав свой скрипт на Python или javascript). Пример графика приведен в описании эксперимента на сайте \*. Сделать выводы.
7. Запустить программу (cache\_example6.cpp, представленную в файлах лабораторной работы в облаке ТПУ) на своем компьютере (или аудиторном ПК). Объяснить, что происходит в программе. Сделать выводы по результатам работы.

**Код программы**

#include <iostream>

#include <cassert>

#include <chrono>

#define ROWS 2

#define COLUMN 10

using namespace std;

using namespace std::chrono;

void CreatingAndChecking(uint64\_t\* size, bool\* isEnoughMem);

int8\_t\*\* Change(int8\_t\*\* m, unsigned char num\_mass, unsigned char num\_of\_change, int8\_t num);

int8\_t\*\* ChangeWithPointer(int8\_t\*\* m, unsigned char num\_mass, unsigned char num\_of\_change, int8\_t num);

int8\_t\*\* InitializeDoubleMas(int8\_t rows, int8\_t columns);

int8\_t\* InitializeMas(int8\_t size);

void ShowMas(int8\_t\* mas, int8\_t size);

void DeleteDoubleMas(int8\_t\*\* mases, int8\_t rows, int8\_t columns);

int main() //mas = new int[2 ^ 1073741824]; // 2 ^ (2 ^ 30 - 1)

{

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

uint64\_t\* len = new uint64\_t;

bool\* isEnoughMem = new bool;

\*len = 3000000000;

\*isEnoughMem = true;

while (\*isEnoughMem)

{

\*len += 2000000;

CreatingAndChecking(len, isEnoughMem);

}

int8\_t\*\* mases = InitializeDoubleMas(ROWS, COLUMN);

cout << "Два изначальных массива" << endl;

ShowMas(mases[0], COLUMN);

ShowMas(mases[1], COLUMN);

int8\_t\*\* mases2 = Change(mases, 0, 20, 100);

int8\_t num = 10;

//assert(num >= 0 && num < COLUMN);

//cout << (int)mases2[1][num] << endl;

cout << endl << "Выход за границы первого массива, меняют случайную переменную во втором" << endl;

ShowMas(mases2[0], COLUMN);

ShowMas(mases2[1], COLUMN);

DeleteDoubleMas(mases2, ROWS, COLUMN);

mases2 = ChangeWithPointer(mases, 0, 20, 100);

cout << endl << "Выход за границы первого массива, меняют случайную переменную во втором (указатели)" << endl;

ShowMas(mases2[0], COLUMN);

ShowMas(mases2[1], COLUMN);

DeleteDoubleMas(mases, ROWS, COLUMN);

DeleteDoubleMas(mases2, ROWS, COLUMN);

int\* Massive = new int[1000000];

char c = ' ';

while (c != 'q') {

cin >> c;

cout << endl;

}

c = ' ';

delete[] Massive;

while (c != 'q') {

cin >> c;

cout << endl;

}

size\_t steps = 64 \* 1024 \* 1024; // Arbitrary number of steps

unsigned int lengthMod = 256;

for (size\_t i = 0; lengthMod < steps; i++)

{

auto start = high\_resolution\_clock::now();

lengthMod = 256 \* pow(2, i);

int32\_t\* arr = new int32\_t[lengthMod]; // (x & lengthMod) is equal to (x % arr.Length)

auto end = high\_resolution\_clock::now();

delete[] arr;

auto duration = duration\_cast<microseconds>(end - start);

cout << "size : " << (lengthMod \* 32) / 8192 << "kb ::" << " time : " << duration.count() << "ms" << endl;

}

c = ' ';

while (c != 'q') {

cin >> c;

cout << endl;

}

return 0;

}

void CreatingAndChecking(uint64\_t\* size, bool\* isEnoughMem)

{

try

{

int\* Massive = new int[\*size];

cout << \*size << endl;

delete[] Massive;

}

catch (...)

{

cout << \*size << " элементов :: " << (\*size \* 32) / (8 \* 1024 \* 1024) << " Mb" << endl;

\*isEnoughMem = false;

}

}

int8\_t\*\* Change(int8\_t\*\* m, unsigned char num\_mass, unsigned char num\_of\_change, int8\_t num) {

int8\_t\*\* mass = InitializeDoubleMas(ROWS, COLUMN);

for (size\_t i = 0; i < ROWS; i++) {

for (size\_t j = 0; j < COLUMN; j++) {

mass[i][j] = m[i][j];

}

}

if ((num\_of\_change >= 0 && num\_of\_change < COLUMN) && (num\_mass >= 0 && num\_mass < ROWS)) {

mass[num\_mass][num\_of\_change] = num;

}

else {

mass[num\_mass ^ 1][rand() % COLUMN] = (int8\_t)0xFF;

}

return mass;

}

int8\_t\*\* ChangeWithPointer(int8\_t\*\* m, unsigned char num\_mass, unsigned char num\_of\_change, int8\_t num) {

int8\_t\*\* mass = InitializeDoubleMas(ROWS, COLUMN);

for (size\_t i = 0; i < ROWS; i++) {

for (size\_t j = 0; j < COLUMN; j++) {

mass[i][j] = m[i][j];

}

}

if ((num\_of\_change >= 0 && num\_of\_change < COLUMN) && (num\_mass >= 0 && num\_mass < ROWS)) {

\*(\*(mass + num\_mass) + num\_of\_change) = num;

}

else {

\*(\*(mass + (num\_mass ^ 1)) + (rand() % 10)) = (int8\_t)0xFF;

}

return mass;

}

int8\_t\*\* InitializeDoubleMas(int8\_t rows, int8\_t columns) {

int8\_t\*\* mases = new int8\_t \* [rows];

for (size\_t i = 0; i < rows; i++) {

mases[i] = new int8\_t[columns];

for (size\_t j = 0; j < columns; j++) {

mases[i][j] = rand() % 255;

}

}

return mases;

}

void DeleteDoubleMas(int8\_t\*\* mases, int8\_t rows, int8\_t columns) {

for (size\_t i = 0; i < rows; i++) {

delete[] mases[i];

}

delete[] mases;

}

int8\_t\* InitializeMas(int8\_t size) {

int8\_t\* mases = new int8\_t [size];

for (size\_t j = 0; j < size; j++) {

mases[j] = rand() % 255;

}

return mases;

}

void ShowMas(int8\_t\* mas, int8\_t size) {

for (size\_t i = 0; i < size; i++) {

if (i != (size - 1)) {

cout << (int)mas[i] << ", ";

}

else {

cout << (int)mas[i] << endl;

}

}

}

**Результат работы программы**

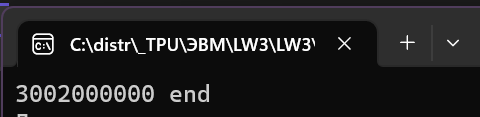
****

Рис. 1. Размер массива, при котором система не даст выделить память (без файлов подкачки)

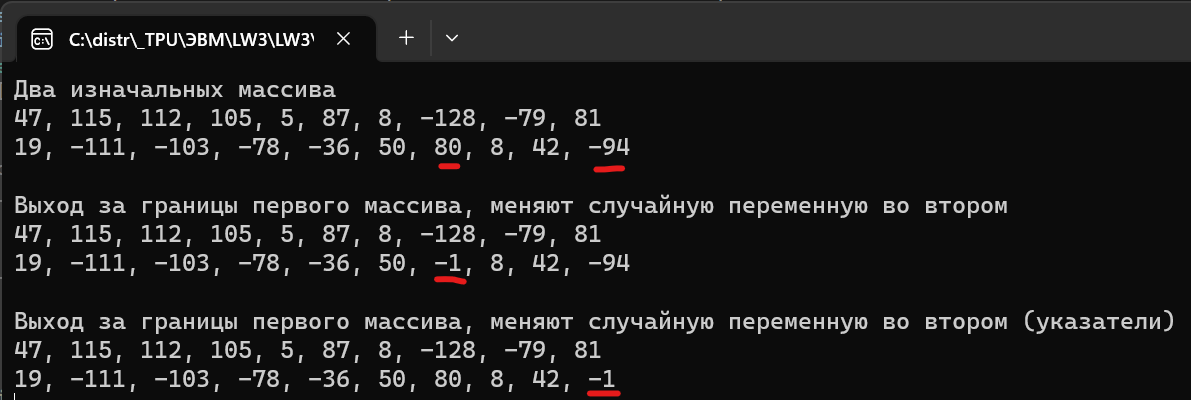
****

Рис. 2. Выход за границы массива

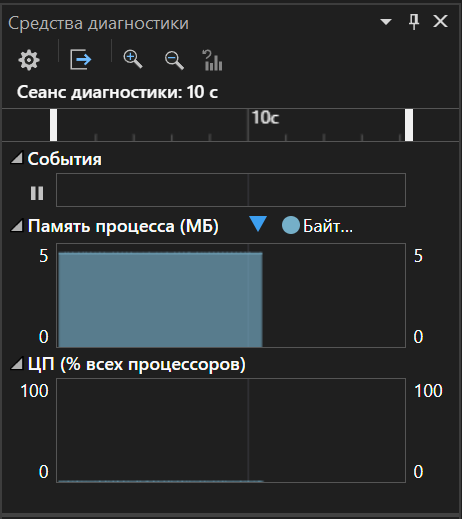
****

Рис. 3. Выделенная память без очистки

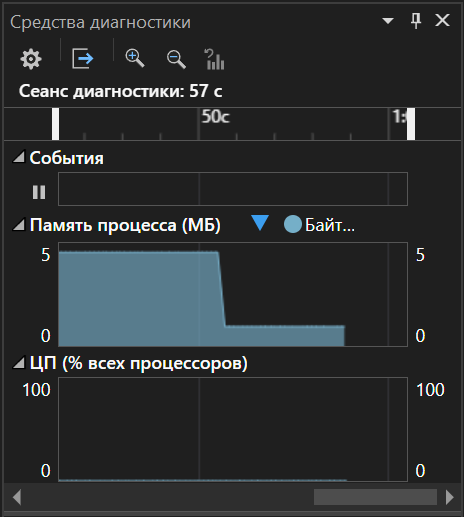
****

Рис. 4. Выделенная память после устранения утечки

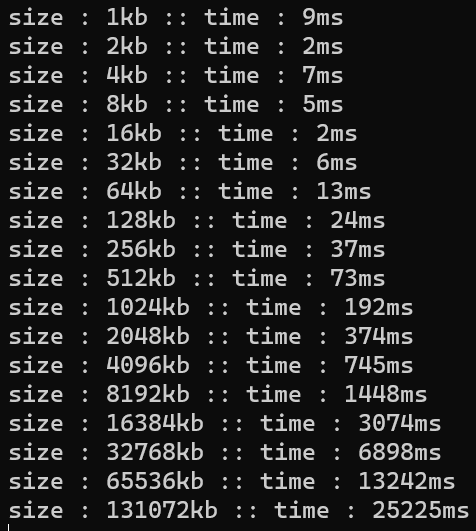
****

Рис. 5. Выделенная память после устранения утечки

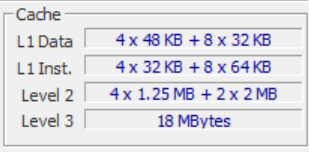


Рис. 6. Размер кэш памяти процессора

Зависимость выделяемой памяти от времени

****

Рис. 7. Результат работы программы “cache\_example6.cpp”

**Вывод**

В ходе лабораторной работы было создано 64 битное приложение C++, в котором было определено максимальное значение динамически выделяемой памяти под массив (для своей системы), была программно воспроизведена ситуация, при которой выход за границы массива искажает значения массива обычным способом и с помощью указателей. Также была продемонстрирована утечка памяти, и результат работы программы “cache\_example6.cpp”.