**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО»**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**ОТЧЕТ**

**по Лабораторной работе № 17**

**«Знакомство с динамическими структурами данных»**

Специальность 09.02.07«Информационные системы и программирование»

Дисциплина «Основы алгоритмов и программирование»

|  |  |
| --- | --- |
| Преподаватель:  Яковлева Н.М.  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_202\_\_г.  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Выполнил:  студент группы Y2231  Петров Н.В. |

Санкт-Петербург

2019/2020

Программа для заполнения стека

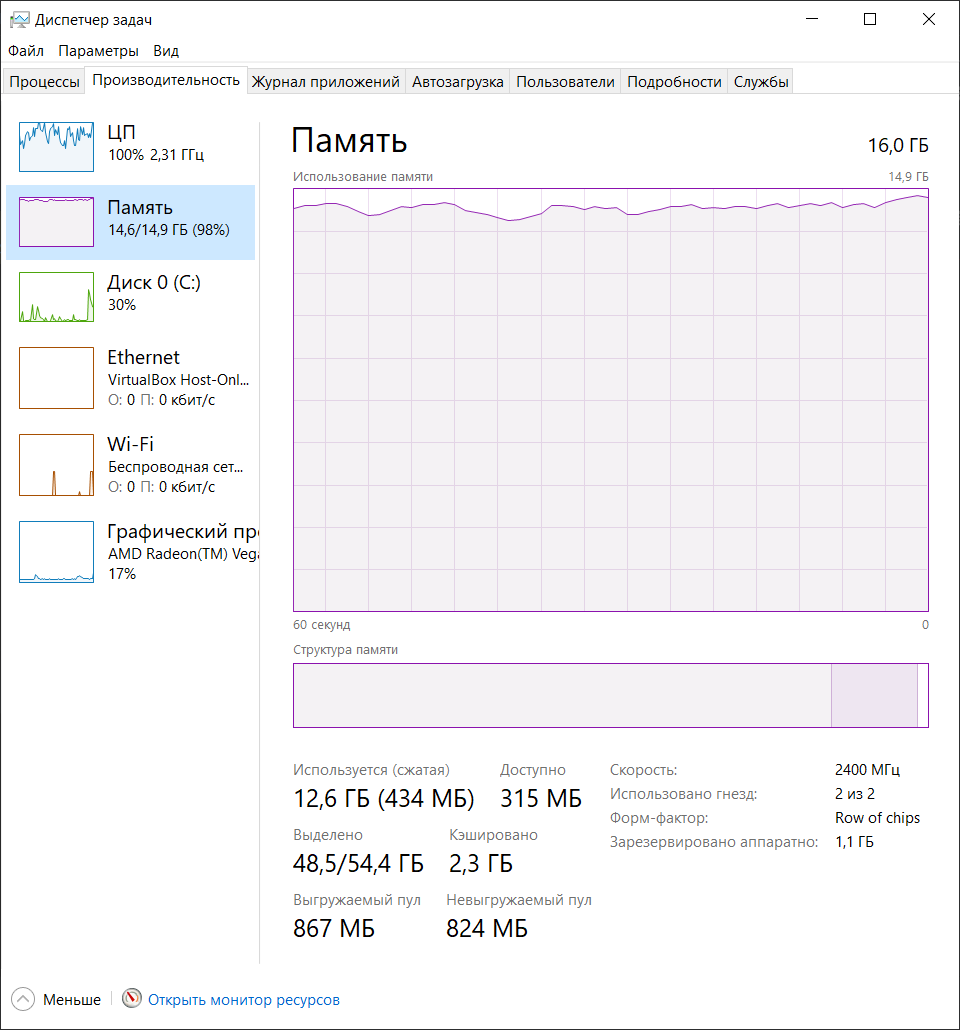
#include <iostream>  
  
int fast\_overflow(long long int d) {  
 return fast\_overflow(d+1);  
}  
  
int slow\_overflow(int d) {  
 std::cout << d << std::endl;  
 return slow\_overflow(d+1);  
}  
  
int main() {  
 std::cout << slow\_overflow(0) << std::endl;  
 return 0;  
}

Первая функция рекурсивно вызывается, и просто быстро забивает стек, а вторая перед рекурсивным вызовом выводит текущее к-во повторений, в моей конфигурации было 262020 повторений \* 4 байта (размер int) = 1 048 080 байт – примерный размер стека.

Программа для заполнения кучи

#include <iostream>  
using namespace std;  
  
int main() {  
 for (int i = 0;; i++) {  
 int \*p = (int\*)malloc(1024);  
 cout << i << endl;  
 }  
 return 0;  
}

Здесь мы каждый раз выделяем под указатель 1024 байта памяти. К сожалению у меня malloc() ушёл в подкачку, и последнее значение было 28018757 (28 691 207 168 байт)



Из вышеописанного следует, что стек намного меньше кучи.

Программа, работающая с динамическим списком в диалоговом режиме

main.cpp

#include <iostream>  
#include "dynamic\_array.h"  
using namespace std;  
  
int main() {  
 short int todo;  
 bool is\_init = false;  
 Node pbeg = Node{0, nullptr, nullptr};  
 Node \*pend = &pbeg;  
 int val;  
 while (true) {  
 cout << "Выберите режим работы:\n"  
 "1. Вывод списка\n"  
 "2. Искать элемент\n"  
 "3. Добавить элемент в конец\n"  
 "4. Вставить элемент\n"  
 "5. Выйти из программы\n";  
  
 cin >> todo;  
 switch (todo) {  
 case 1:  
 print(&pbeg);  
 break;  
 case 2:  
 if (is\_init) {  
 cout << "Введите искомое значение: ";  
 cin >> val;  
 Node \*pf = find(&pbeg, val);  
 if (pf) cout << pf->d << endl;  
 else cout << "Не найдено!" << endl;  
 } else {  
 cout << "Список пуст!" << endl;  
 }  
 break;  
 case 3:  
 cout << "Введите добавляемое значение: ";  
 cin >> val;  
 if (is\_init) {  
 add(&pend, val);  
 } else {  
 pbeg.d = val;  
 is\_init = true;  
 }  
 break;  
 case 4:  
 cout << "Введите добавляемое значение: ";  
 cin >> val;  
 int key;  
 cout << "Введите номер элемента: ";  
 cin >> key;  
 if (is\_init) {  
 insert(&pbeg, &pend, val, key);  
 } else {  
 pbeg.d = val;  
 is\_init = true;  
 }  
 break;  
 default:  
 case 5:  
 return 0;  
 }  
 }  
}

dynamic\_array.h

#ifndef LAB17V3\_DYNAMIC\_ARRAY\_H  
#define LAB17V3\_DYNAMIC\_ARRAY\_H  
  
struct Node {  
 int d;  
 Node \*next;  
 Node \*prev;  
};  
  
void print(Node \*const array\_start);  
void add(Node \*\*pend, int d);  
Node \*find(Node \*const pbeg, int key);  
bool remove(Node \*\*pbeg, Node \*\*pend, int key);  
Node \*insert(Node \*const pbeg, Node \*\*pend, int key, int d);  
  
#endif //LAB17V3\_DYNAMIC\_ARRAY\_H

dynamic\_array.cpp

#include <iostream>  
#include "dynamic\_array.h"  
  
using namespace std;  
  
void print(Node \*const array\_start) {  
 Node \*tmp\_pointer = array\_start;  
 int len = 1;  
 while (tmp\_pointer) {  
 tmp\_pointer = tmp\_pointer->next;  
 len++;  
 }  
 int vals[--len];  
 tmp\_pointer = array\_start;  
 for (int i = 0; i < len; i++) {  
 vals[i] = tmp\_pointer->d;  
 tmp\_pointer = tmp\_pointer->next;  
 }  
 bool is\_sorted = false;  
 while (!is\_sorted) {  
 is\_sorted = true;  
 for (int i = 0; i < len - 1; i++) {  
 if (vals[i] > vals[i + 1]) {  
 is\_sorted = false;  
 int tmp = vals[i];  
 vals[i] = vals[i + 1];  
 vals[i + 1] = tmp;  
 }  
 }  
 }  
 for (const auto &e : vals) {  
 cout << e << " ";  
 }  
 cout << endl;  
}  
  
void add(Node \*\*pend, int d) {  
 Node \*pv = new Node;  
 pv->d = d;  
 pv->next = nullptr;  
 pv->prev = \*pend;  
 (\*pend)->next = pv;  
 \*pend = pv;  
}  
  
Node \*find(Node \*const pbeg, int key) {  
 Node \*pv = pbeg;  
 while (pv) {  
 if (pv->d == key) break;  
 pv = pv->next;  
 }  
 return pv;  
}  
  
bool remove(Node \*\*pbeg, Node \*\*pend, int key) {  
 if (Node \*pkey = find(\*pbeg, key)) {  
 if (pkey == \*pbeg) {  
 \*pbeg = (\*pbeg)->next;  
 (\*pbeg)->prev = nullptr;  
 } else if (pkey == \*pend) {  
 \*pend = (\*pend)->prev;  
 (\*pend)->next = nullptr;  
 } else {  
 (pkey->prev)->next = pkey->next;  
 (pkey->next)->prev = pkey->prev;  
 }  
 delete pkey;  
 return true;  
 }  
 return false;  
}  
  
Node \*insert(Node \*const pbeg, Node \*\*pend, int key, int d) {  
 if (Node \*pkey = find(pbeg, key)) {  
 Node \*pv = new Node;  
 pv->d = d;  
 pv->next = pkey->next;  
 pv->prev = pkey;  
 pkey->next = pv;  
 if (pkey != \*pend)  
 (pv->next)->prev = pv;  
 else  
 \*pend = pv;  
 return pv;  
 }  
 return nullptr;  
}

Тестирование: (единственная написанная мною функция – вывод с сортировкой)

Выберите режим работы:  
1. Вывод списка  
2. Искать элемент  
3. Добавить элемент в конец  
4. Вставить элемент  
5. Выйти из программы  
3  
Введите добавляемое значение: 7  
Выберите режим работы:  
1. Вывод списка  
2. Искать элемент  
3. Добавить элемент в конец  
4. Вставить элемент  
5. Выйти из программы  
3  
Введите добавляемое значение: 6  
Выберите режим работы:  
1. Вывод списка  
2. Искать элемент  
3. Добавить элемент в конец  
4. Вставить элемент  
5. Выйти из программы  
3  
Введите добавляемое значение: 5  
Выберите режим работы:  
1. Вывод списка  
2. Искать элемент  
3. Добавить элемент в конец  
4. Вставить элемент  
5. Выйти из программы  
1  
5 6 7  
Выберите режим работы:  
1. Вывод списка  
2. Искать элемент  
3. Добавить элемент в конец  
4. Вставить элемент  
5. Выйти из программы  
5

Мусор в памяти

Мусор в памяти при работе с динамическими массивами может быть получен в следующих ситуациях:

Если мы “выбрасываем” элемент из середины списка, методом перевода указателя элемента перед ним, на элемент за ним. В таком случае в памяти представление этого элемента не меняется, и он висит “мусором”

Если мы сокращаем массив, удаляя первый или последний элемент тем же образом (удаляя на них ссылки). Мусор образовывается из-за того, что мы при этом оставляем их, и не перезаписываем.

Если мы создаем ещё раз переменную в динамической памяти с тем же названием, то у нас происходит утечка памяти и образуется мусор, т.к. динамическая память высвобождается только при явном вызове delete или в конце программы.