Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №2 дисциплины «Технологии программирования»

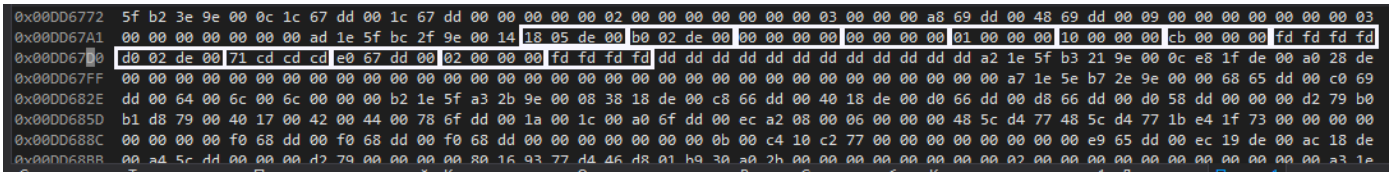
Выполнил студент группы ИВТ-22_____/Крючков И. С/
Проверил_____/Долженкова М. Л./

Киров 2022

1. Задание

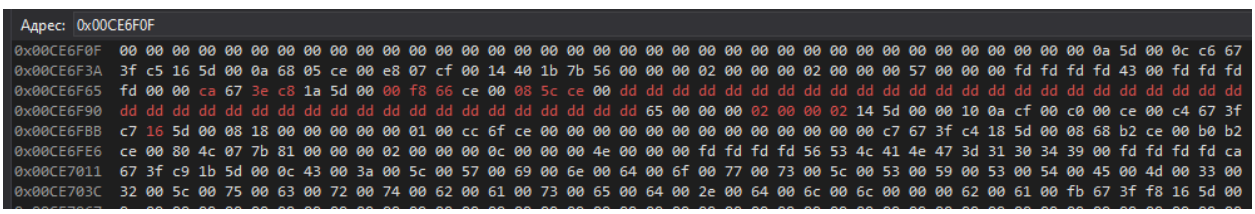
Написать программу для работы с динамической структурой данных – циклическая очередь, содержащей массив целых чисел и символ. Изучить дамп памяти. Для работы с памятью использовать realloc.

2. Структура дампа памяти



- Указатель на предыдущий занятый элемент
- Указатель на следующий занятный элемент
- Указатель на имя файла подкачки
- Номер строки в файле подкачки
- Размер выделенного участка памяти
- Тип участка памяти
- Количество обращений
- Индикатор начала участка памяти
- Указатель на массив
- Символ
- Указатель на следующий элемент
- Размер массива
- Индикатор конца участка памяти

3. Освобождения памяти при удалении элемента



4. Краткие теоретические сведения

Функция realloc() изменяет величину выделенной памяти, на которую указывает передаваемый первым параметром указатель, на новую величину,

задаваемую вторым параметром, которая задается в байтах и может быть больше или меньше оригинала. Возвращается указатель на блок памяти, поскольку может возникнуть необходимость переместить блок при возрастании его размера. В таком случае содержимое старого блока копируется в новый блок и информация не теряется.

Если свободной памяти недостаточно для выделения в куче блока нового рамера, то возвращается нулевой указатель.

5. Листинг программы

```
#include <iostream>
#include <limits>

using namespace std;

struct QueueElement {
    int* masInt;
    char symbol;
   QueueElement* next;
   int arrSize;
};



int get_int( int min, int max) {
   int v;
   while (true) {
         if ((cin >> v).good()) {
               if (v >= min and v <= max) {
                     std::cin.clear();
                     cin.ignore(numeric_limits<streamsize>::max(), '\n');
                     return v;
               }
               else {
                     cout << "Invalid value" << endl;
                     cin.clear();
                     cin.ignore(numeric_limits<streamsize>::max(), '\n');
                     continue;
               }
         }
```

```
		else {
			cout << "Invalid value" << endl;
			cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');
			continue;
		}
	}
}


char get_symbol() {
	char t;
	while (true) {
		if ((cin >> t).good()) {
			std::cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');
			return t;
		}
		else {
			cout << "Invalid value" << endl;
			cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');
			continue;
		}
	}
}

void pause() {
	if (cin.rdbuf()->in_avail() > 0) {
		cin.clear();
		cin.ignore(numeric_limits<streamsize>::max(), '\n');
	}
	cin.get();
}

void change_links(QueueElement* q, int c) {
	for (int i = 0; i < c; i++) {
		if (i == c - 1) {
			(q + i)->next = q;
		}
		else {
```

```
                (q + i)->next = (q + i + 1);
            }
        }
    }

    int main()
    {

        int n = 0; // номер команды
        int count = 0; // кол-во элементов в очереди
        QueueElement* quenue = NULL;
        QueueElement* quenue_tmp;
        QueueElement* quenue_z;


        do {
            if (cin.rdbuf()->in_avail() > 0) {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
            }
            system("cls");
            cout << "Commands:" << endl
                << "1 - add element" << endl
                << "2 - display elements" << endl
                << "3 - delete element" << endl
                << "4 - clear quenue" << endl
                << "5 - exit" << endl;

            n = get_int(1, 5);

            switch (n)
            {
                case 1: {
                    if (count == 0) {
                        quenue = (QueueElement*)realloc(NULL,
sizeof(QueueElement));
                        cout << "Number of array elements:" << endl;
                        quenue->arrSize = get_int(1, 10);
                        quenue->masInt = (int*)realloc(NULL,
sizeof(int)* quenue->arrSize);
```

```
                                    for (int i = 0; i < queue->arrSize; i++) {
                                          cout << "M[" << i << "] = ";
                                          queue->masInt[i] = get_int(-2147483648,
2147483647);
                                    }

                                    cout << "Enter char: " << endl;
                                    queue->symbol = get_symbol();
                                    queue->next = (queue + 1);
                                    count++;
                              }
                              else {
                                    queue_tmp    =    (QueueElement*)realloc(NULL,
sizeof(*queue)+sizeof(QueueElement) * count);
                                    memcpy(queue_tmp,  queue,  sizeof(*queue)  +
sizeof(QueueElement) * (count-1));

                                    change_links(queue_tmp, count);
                                    realloc(queue, 0);

                                    cout << "Number of array elements:" << endl;
                                    (queue_tmp + count)->arrSize = get_int(1, 10);
                                    (queue_tmp      +      count)->masInt      =
(int*)realloc(NULL, sizeof(int) * (queue_tmp + count)->arrSize);

                                    for  (int  i  =  0;  i  <  (queue_tmp  +  count)-
>arrSize; i++) {
                                          cout << "M[" << i << "] = ";
                                          (queue_tmp   +   count)->masInt[i]   =
get_int(-2147483648, 2147483647);
                                    }

                                    cout << "Enter char: " << endl;
                                    (queue_tmp + count)->symbol = get_symbol();

                                    (queue_tmp + count - 1)->next = (queue_tmp +
count);
                                    (queue_tmp + count)->next = queue_tmp;
                                    cout << sizeof(*queue) << endl;
                                    cout << sizeof(*queue_tmp) << endl;
                                    queue = queue_tmp;
```

```
                                    count++;
                                }

                                break;
                            }
                            case 2: {
                                if (count != 0) {
                                    queue_z = queue;
                                    for (int i = 0; i < count; i++) {
                                        cout << "#" << i << ":" << endl;
                                        cout << "Array: ";
                                        for (int j = 0; j < queue_z->arrSize;
j++) {
                                            cout << queue_z->masInt[j] << "
";
                                        }
                                        cout << endl;

                                        cout << "Char: " << queue_z->symbol <<
endl;

                                        queue_z = queue_z->next;
                                    }
                                }
                                else {
                                    cout << "Empty" << endl;
                                }
                                pause();
                                break;
                            }
                            case 3: {
                                if (count != 0) {
                                    queue_tmp    =    (QueueElement*)realloc(NULL,
sizeof(QueueElement) * (count-1));
                                    memcpy(queue_tmp,                queue->next,
sizeof(QueueElement) * (count - 1));
                                    change_links(queue_tmp, count-1);
                                    realloc(queue->masInt, 0);
                                    realloc(queue, 0);

                                    queue = queue_tmp;
```

```
                                count--;
                                cout << "Deleted" << endl;
                        }
                        else {
                                cout << "Empty" << endl;
                        }
                        pause();
                        break;
                }
                case 4: {
                        if (count != 0) {
                                quenue_z = quenue;
                                for (int i = 0; i < count; i++) {
                                        realloc(quenue_z->masInt, 0);
                                        quenue_z = quenue_z->next;
                                }

                                realloc(quenue, 0);
                                quenue = NULL;
                                count = 0;

                                cout << "Deleted" << endl;
                        }
                        else {
                                cout << "Empty" << endl;
                        }
                        pause();
                        break;
                }
                default:
                        break;
                }
        } while (n != 5);

         return 0;
}
```

6. Вывод

В ходе выполнения лабораторной работы была разработана структура данных на динамической памяти – циклическая очередь, для работы с памятью был применен realloc. Изучен дамп памяти при выполнении

различных операций со структурой.