МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«**Вятский государственный университет**»

**(«ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №2

по дисциплине «Технологии программирования»

Выполнил студент группы ИВТ-21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Щесняк Д. С./

Проверил доцент кафедры ЭВМ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М. Л./

Киров 2016

1. Задание на лабораторную работу

Используя функции malloc и free реализовать однонаправленный кольцевой список, который содержит внутри себя указатель на массив и 3 символа. Проанализировать дамп памяти и на основе этого сделать вывод о распределении памяти в методе граничных маркеров.

1. Исследование дампа памяти

Дамб блока памяти, в которой содержится элемент списка представлен на рисунке 1



Рисунок 1 – дамп элемента списка

Между граничных маркеров fdfdfdfd заключены данные. Первые четыре байта указывают на адрес следующего элемента списка (0x008B9610). Так как элемент в списке 1, то он указывает сам на себя. Следующие 4 байта указывают на положение в памяти массива (0x008B73B0). И последние 4 байта содержат строку состоящую из 3 символов, 4-й байт необходим для выравнивания. Перед началом блока fdfdfdfd указана служебная информация, которая содержит в себе размер структуры (12 байт), сколько переменных ссылаются на этот блок, количество использований.

Дамп памяти после добавления элемента представлен на рисунке 2.



Рисунок 2 – дамп памяти после добавления элемента

Как мы видим, после добавление элемента, первые 4 байта, которые указывают на следующий элемент, изменились (0x008B8920). Если перейти по этому адресу, то можно увидеть похожий блок памяти. Дамп памяти второго элемента представлен на рисунке 3



Рисунок 3 – дамп памяти второго элемента

После удаления первого элемента списка, весь блок памяти пометился как свободный, это представлено на рисунке 4.

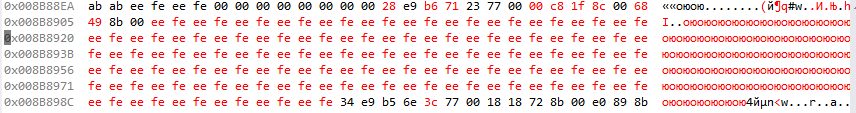


Рисунок 4 – освобождение памяти

1. Исходный код программы

Исходный код программы представлен на рисунке 1.

|  |
| --- |
| #include <cstdlib>  #include <iostream>  #include <string>  #include <locale.h>  #include <conio.h>  using namespace std;  const string mainMenuText = "1 - Добавить элемент\n2 - Удалить элемент\n3 - Вывести структуру\n\n0 - Выход";  const string addElemText = "Первым аргументом введите место куда добавить\nВторым аргументом - строку из 3-х символов";  const string delElementText = "Введите номер удаляемого элемента";  const string printText = "Структура содержит: ";  const string emptyText = "Структура пустая";  struct elem  {  elem\* pnext;  int\* m;  char c[3];  };  class List  {  public:  elem\* first;  List() {  first = NULL;  }  List(char word[3]) {  first = (elem\*) malloc(sizeof(elem));  first->pnext = first;  first->m = (int\*) malloc(sizeof(int)\*3);  for (int i = 0; i < 3; i++)  {  first->c[i] = word[i];  }  }  void add(int pos, const char word[3]) {  elem\* n = (elem\*) malloc(sizeof(elem));  n->m = (int\*) malloc(sizeof(int) \* 3);  for (int i = 0; i < 3; i++) {  n->c[i] = word[i];  }  if (pos == 1) {  elem\* p = first;  while (p->pnext != first) {  p = p->pnext;  }  p->pnext = n;  n->pnext = first;  first = n;  } else {  elem\* p = first;  for (int i = 1; i < pos - 1; i++)  {  p = p->pnext;  }  n->pnext = p->pnext;  p->pnext = n;  }  }  bool del(int n) {  elem\* p = first;    for(int i = 1; i < n; i++) {  p = p->pnext;  }  elem\* prev = first;  while (prev->pnext != p) {  prev = prev->pnext;  }    prev->pnext = p->pnext;  if (prev == p) {  first = NULL;  }  free(p->m);  free(p);  if (prev->pnext == prev) {  first = prev;  }  return true;  }  void print() {  elem\* p = first;  do  {  for (int i = 0; i < 3; i++)  {  cout << p->c[i];  }  cout << endl;  p = p->pnext;  } while (p != first);  }  };  int main(int atgc, char\* arg[]) {  setlocale(LC\_ALL, "RUS");  char key = 1;  char word[3];  List l;  do  {  system("cls");  cout << mainMenuText << endl;  key = \_getch();  system("cls");  switch (key)  {  case '1': {  if (l.first == NULL) {  cout << "Введите 3 символа для первого элемента" << endl;  for (int i = 0; i < 3; i++)  {  cin >> word[i];  }  l = List(word);  } else {  int n;  cout << addElemText << endl;  cin >> n;    for (int i = 0; i < 3; i++)  {  cin >> word[i];  }  l.add(n, word);  cout << "Элемент добавлен на позицию " << n << endl;  }  system("pause");  break;  };  case '2': {  if (l.first == NULL) {  cout << emptyText << endl;  } else {  int n;  cout << delElementText << endl;  cin >> n;  l.del(n);  }  system("pause");  break;  };  case '3': {  if (l.first == NULL) {  cout << emptyText << endl;  } else {  cout << printText << endl;  l.print();  }  system("pause");  break;  };  default:  break;  }  } while (key != '0');  return 0;  } |

Рисунок 1 – Исходный код программы

1. Вывод

В ходе выполнения лабораторной работы были полученные необходимые знания базового синтаксиса языка программирования С++. Была изучена организация памяти методом граничных маркеров. Данные знания являются фундаментальными и необходимы для дальнейшего продолжения изучения различных технологий программирования и языка С++.