Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение

высшего образования

«**Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные

системы»

**ОТЧЁТ**

**по лабораторной работе №18.6**

Дисциплина: «Информатика»

Тема: “Объектно-ориентированное программирование.

АДФ. Контейнеры.”

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

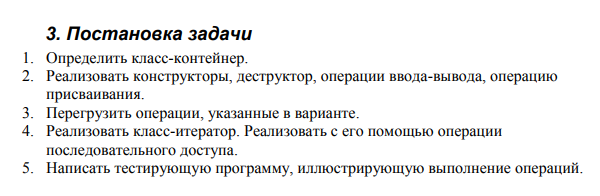
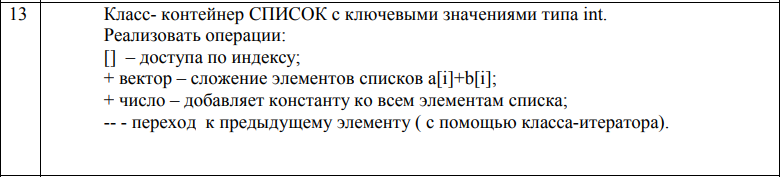
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**

****

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Организовать класс List.
   2. Организовать класс Iterator, дружественный классу List
   3. Организовать перегрузку оператора [] для класса List.
   4. Организовать перегрузку оператора <<, дружественной классу List.
   5. Организовать перегрузку оператора ++ для класса Iterator.
   6. Организовать перегрузку оператора -- для класса Iterator.
   7. Организовать перегрузку оператора == для класса Iterator.
   8. Организовать перегрузку оператора != для класса Iterator.
   9. Организовать перегрузку оператора \* для класса Iterator.
2. В ходе работы были использованы следующие типы данных:
   1. Для класса List описаны следующие поля:
3. class List
4. {
5. int\* data;
6. int size;
7. Iterator \_begin;
8. Iterator \_end;
   * 1. Конструкторы класса List описаны в файле List.cpp.
9. List::List(int s, int k)
10. {
11. size = s;
12. data = new int[size];
13. for (int i = 0; i < size; i++) data[i] = k;
14. \_begin.elem = &data[0];
15. \_end.elem = &data[size];
16. }
18. List::List(const List& a)
19. {
20. size = a.size;
21. data = new int[size];
22. for (int i = 0; i < size; i++) data[i] = a.data[i];
23. \_begin.elem = &data[0];
24. \_end.elem = &data[size];
25. }
    * 1. Деструктор класса List
26. List::~List()
27. {
28. delete[] data;
29. data = 0;
30. }
    * 1. Перегрузка оператора присваивания
31. List& List::operator=(const List& a)
32. {
33. if (this == &a) return \*this;
34. size = a.size;
35. if (data != 0) delete[] data;
36. data = new int[size];
37. for (int i = 0; i < size; i++) data[i] = a.data[i];
38. \_begin.elem = &data[0];
39. \_end.elem = &data[size];
40. return \*this;
41. }
    * 1. Перегрузка оператора [] для класса List
42. int& List::operator[](int index)
43. {
44. if (index < size) return data[index];
45. else cout << "\nError! Index out of range.";
46. }
    * 1. Перегрузка оператора +
47. List List::operator+(const int k)
48. {
49. List temp(size);
50. for (int i = 0; i < size; ++i) temp.data[i] += data[i] + k;
51. return temp;
52. }
54. List List::operator+(const List& a)
55. {
56. int min = size < a.size ? size : a.size;
57. List temp(min);
58. for (int i = 0; i < min; i++) temp.data[i] = data[i] + a.data[i];
59. return temp;
60. }
    * 1. Перегрузка оператора ()
61. int List::operator () ()
62. {
63. return size;
64. }
    * 1. Перегрузка операторов >> и <<
65. ostream& operator<<(ostream& out, List& a)
66. {
67. for (int i = 0; i < a(); ++i) out << a.data[i] << " ";
68. return out;
69. }
71. istream& operator>>(istream& in, List& a)
72. {
73. for (int i = 0; i < a(); ++i) in >> a.data[i];
74. return in;
75. }
    1. Для класса Iterator описаны следующие поля:
76. class Iterator
77. {
78. friend class List;
79. int\* elem;
    * 1. Конструкторы класса Iterator
80. Iterator() { elem = 0; }
81. Iterator(const Iterator& a) { elem = a.elem; }
    * 1. Перегрузка операций для класса Iterator
82. bool operator == (const Iterator& a) { return elem == a.elem; }
83. bool operator != (const Iterator& a) { return elem != a.elem; }
84. void operator ++ () { ++elem; }
85. void operator ++ (int) { elem++; }
86. void operator -- () { --elem; }
87. void operator -- (int) { elem--; }
88. void operator + (const int k) { elem += k; }
89. void operator - (const int k) { elem -= k; }
90. int& operator \* () { return \*elem; }

**Блок-схема программы**

**Код программы**

**Файл List.h:**

1. #pragma once
2. #include <iostream>
3. using namespace std;
5. class Iterator
6. {
7. friend class List;
8. int\* elem;
9. public:
10. Iterator() { elem = 0; }
11. Iterator(const Iterator& a) { elem = a.elem; }
13. bool operator == (const Iterator& a) { return elem == a.elem; }
14. bool operator != (const Iterator& a) { return elem != a.elem; }
15. void operator ++ () { ++elem; }
16. void operator ++ (int) { elem++; }
17. void operator -- () { --elem; }
18. void operator -- (int) { elem--; }
19. void operator + (const int k) { elem += k; }
20. void operator - (const int k) { elem -= k; }
21. int& operator \* () { return \*elem; }
22. };
24. class List
25. {
26. int\* data;
27. int size;
28. Iterator \_begin;
29. Iterator \_end;
30. public:
31. List(int s, int k = 0);
32. List(const List& a);
33. ~List();
35. List& operator = (const List& a);
36. int& operator [] (int index);
37. List operator + (const int k);
38. List operator + (const List& a);
39. int operator () ();
41. friend ostream& operator << (ostream& out, List& a);
42. friend istream& operator >> (istream& in, List& a);
44. Iterator first() { return \_begin; }
45. Iterator last() { return \_end; }
46. };

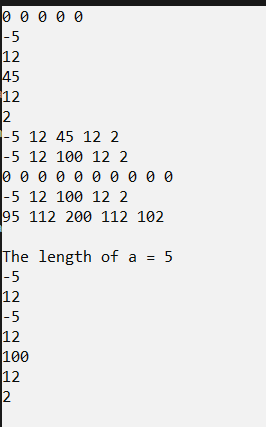
**Файл List.cpp:**

1. #pragma once
2. #include "List.h"
4. List::List(int s, int k)
5. {
6. size = s;
7. data = new int[size];
8. for (int i = 0; i < size; i++) data[i] = k;
9. \_begin.elem = &data[0];
10. \_end.elem = &data[size];
11. }
13. List::List(const List& a)
14. {
15. size = a.size;
16. data = new int[size];
17. for (int i = 0; i < size; i++) data[i] = a.data[i];
18. \_begin.elem = &data[0];
19. \_end.elem = &data[size];
20. }
22. List::~List()
23. {
24. delete[] data;
25. data = 0;
26. }
28. List& List::operator=(const List& a)
29. {
30. if (this == &a) return \*this;
31. size = a.size;
32. if (data != 0) delete[] data;
33. data = new int[size];
34. for (int i = 0; i < size; i++) data[i] = a.data[i];
35. \_begin.elem = &data[0];
36. \_end.elem = &data[size];
37. return \*this;
38. }
40. int& List::operator[](int index)
41. {
42. if (index < size) return data[index];
43. else cout << "\nError! Index out of range.";
44. }
46. List List::operator+(const int k)
47. {
48. List temp(size);
49. for (int i = 0; i < size; ++i) temp.data[i] += data[i] + k;
50. return temp;
51. }
53. List List::operator+(const List& a)
54. {
55. int min = size < a.size ? size : a.size;
56. List temp(min);
57. for (int i = 0; i < min; i++) temp.data[i] = data[i] + a.data[i];
58. return temp;
59. }
61. int List::operator () ()
62. {
63. return size;
64. }
66. ostream& operator<<(ostream& out, List& a)
67. {
68. for (int i = 0; i < a(); ++i) out << a.data[i] << " ";
69. return out;
70. }
72. istream& operator>>(istream& in, List& a)
73. {
74. for (int i = 0; i < a(); ++i) in >> a.data[i];
75. return in;
76. }

**Файл main.cpp:**

1. #include <iostream>
2. #include "List.h"
3. #include <ctime>
4. using namespace std;
6. int main()
7. {
8. setlocale(LC\_ALL, "Russian");
9. srand(time(NULL));
11. List a(5);
12. cout << a << endl;
13. cin >> a;
14. cout << a << endl;
15. a[2] = 100;
16. cout << a << endl;
17. List b(10);
18. cout << b << endl;
19. b = a;
20. cout << b << endl;
21. List c(10);
22. c = b + 100;
23. cout << c << endl;
24. cout << "\nThe length of a = " << a() << endl;
26. cout << \*(a.first()) << endl;
27. Iterator i = a.first();
28. i++;
29. cout << \*i << endl;
31. for (i = a.first(); i != a.last(); i++) cout << \*i << endl;
33. return 0;
34. }

**Скриншоты результатов работы программы**

****

**Ответы на вопросы**

