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 №18.9**

Дисциплина: «Информатика»

Тема: “Объектно-ориентированное программирование.

Обработка исключительных ситуаций.”

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

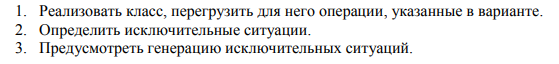
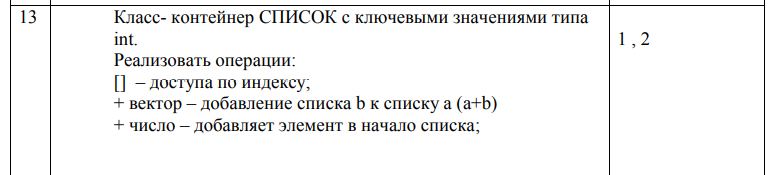
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**

****

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Использовать класс List из ЛР7
   2. Добавить обработку исключительных ситуаций
2. В ходе работы были использованы следующие типы данных:
   1. Класс List, реализацию см. в ЛР7
   2. Класс error с полем str типа string для хранения строки ошибки в заголовочном файле error.h.
3. Для решения задачи данные были представлены в следующем виде:
   1. Данные инициализируются в конструкторе с параметром в классе List
4. Vector::Vector(int s)
5. {
6. if (s > MAX\_SIZE) throw error("Vector length more than MAX\_SIZE\n");
7. size = s;
8. beg = new int[s];
9. for (int i = 0; i < size; i++) beg[i] = 0;
10. }

 Добавлена проверка на случай, если введёный пользователем размер массива больше, чем допустимо возможный.

1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Для вывода массива класса List в консоль используется перегрузка оператора <<
2. ostream& operator << (ostream& out, const Vector& v)
3. {
4. if (v.size == 0) out << "Empty" << endl;
5. else
6. {
7. for (int i = 0; i < v.size; i++) out << v.beg[i] << " ";
8. out << endl;
9. }
10. return out;
11. }

**Блок-схема программы**

**Код программы**

**Реализация первым способом**

**Файл Vector.h:**

1. #pragma once
2. #include <iostream>
3. using namespace std;
5. const int MAX\_SIZE = 30;
7. class Vector
8. {
9. int size;
10. int\* beg;
11. public:
12. Vector() { size = 0; beg = 0; }
13. Vector(int s);
14. Vector(int s, int\* mas);
15. Vector(const Vector& v);
16. ~Vector();
17. const Vector& operator = (const Vector& v);
18. int operator [] (int i);
19. Vector operator + (int a);
20. Vector operator -- ();
22. friend ostream& operator << (ostream& out, const Vector& v);
23. friend istream& operator >> (istream& in, Vector& v);
24. };

**Файл Vector.cpp:**

1. #include "Vector.h"
3. Vector::Vector(int s)
4. {
5. if (s > MAX\_SIZE) throw 1;
6. size = s;
7. beg = new int[s];
8. for (int i = 0; i < size; i++) beg[i] = 0;
9. }
10. Vector::Vector(int s, int\* mas)
11. {
12. if (s > MAX\_SIZE) throw 1;
13. size = s;
14. beg = new int[size];
15. for (int i = 0; i < size; i++) beg[i] = mas[i];
16. }
17. Vector::Vector(const Vector& v)
18. {
19. size = v.size;
20. beg = new int[size];
21. for (int i = 0; i < size; i++) beg[i] = v.beg[i];
22. }
23. Vector::~Vector()
24. {
25. if (beg != 0) delete[] beg;
26. }
27. const Vector& Vector::operator = (const Vector& v)
28. {
29. if (this == &v) return \*this;
30. if (beg != 0) delete[] beg;
31. size = v.size;
32. beg = new int[size];
33. for (int i = 0; i < size; i++) beg[i] = v.beg[i];
34. return \*this;
35. }
36. ostream& operator << (ostream& out, const Vector& v)
37. {
38. if (v.size == 0) out << "Empty\n";
39. else
40. {
41. for (int i = 0; i < v.size; i++) out << v.beg[i] << " ";
42. out << endl;
43. }
44. return out;
45. }
46. istream& operator >> (istream& in, Vector& v)
47. {
48. for (int i = 0; i < v.size; i++)
49. {
50. cout << ">";
51. in >> v.beg[i];
52. }
53. return in;
54. }
55. int Vector::operator [] (int i)
56. {
57. if (i < 0) throw 2;
58. if (i >= size) throw 3;
59. return beg[i];
60. }
61. Vector Vector::operator + (int a)
62. {
63. if (size + 1 == MAX\_SIZE) throw 4;
64. Vector temp(size + 1, beg);
65. temp.beg[size] = a;
66. size++;
67. return temp;
68. }
69. Vector Vector::operator -- ()
70. {
71. if (size == 0) throw 5;
72. if (size == 1)
73. {
74. size = 0;
75. delete[] beg;
76. beg = 0;
77. return \*this;
78. }
79. Vector temp(size, beg);
80. delete[] beg;
81. size--;
82. beg = new int[size];
83. for (int i = 0; i < size; i++) beg[i] = temp.beg[i];
84. return \*this;
85. }

**Файл main.cpp:**

1. #include "Vector.h"
2. #include <iostream>
3. using namespace std;
5. int main()
6. {
7. try
8. {
9. Vector x(2);
10. Vector y;
11. cout << x;
12. cout << "Number? ";
13. int i; cin >> i;
14. cout << x[i] << endl;
15. y = x + 3;
16. cout << y;
17. --x;
18. cout << x;
19. --x;
20. cout << x;
21. --x;
22. }
23. catch (int)
24. {
25. cout << "ERROR!!!" << endl;
26. }
27. return 0;
28. }

**Реализация вторым способом**

**Файл error.h:**

1. #pragma once
2. #include <string>
3. #include <iostream>
4. using namespace std;
6. class error
7. {
8. string str;
9. public:
10. error(string s) { str = s; }
11. void what() { cout << str << endl; }
12. };

**Файл Vector.h:**

1. #pragma once
2. #include <iostream>
3. using namespace std;
5. class Vector
6. {
7. int size;
8. int\* beg;
9. public:
10. Vector() { size = 0; beg = 0; }
11. Vector(int s);
12. Vector(int s, int\* mas);
13. Vector(const Vector& v);
14. ~Vector();
15. const Vector& operator = (const Vector& v);
16. int operator[] (int i);
17. Vector operator + (int i);
18. Vector operator -- ();
19. friend ostream& operator << (ostream& out, const Vector& v);
20. friend istream& operator >> (istream& in, Vector& v);
21. };

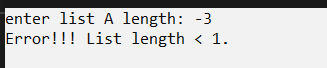
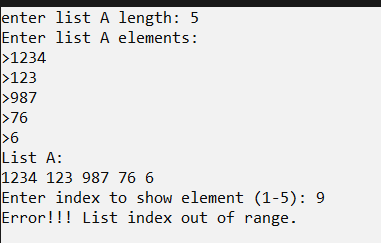
**Файл Vector.cpp:**

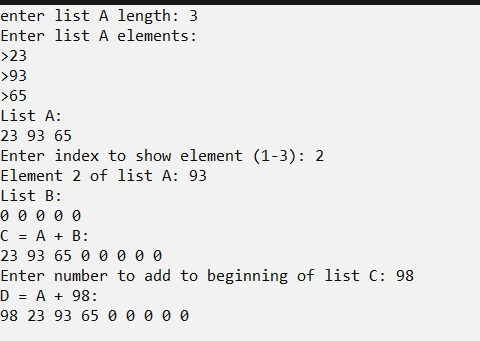
1. #include "Vector.h"
2. #include "Error.h"
3. #include <iostream>
4. using namespace std;
6. const int MAX\_SIZE = 20;
8. Vector::Vector(int s)
9. {
10. if (s > MAX\_SIZE) throw error("Vector length more than MAX\_SIZE\n");
11. size = s;
12. beg = new int[s];
13. for (int i = 0; i < size; i++) beg[i] = 0;
14. }
15. Vector::Vector(int s, int\* mas)
16. {
17. if (s > MAX\_SIZE) throw error("Vector length more than MAX\_SIZE\n");
18. size = s;
19. beg = new int[size];
20. for (int i = 0; i < size; i++) beg[i] = mas[i];
21. }
22. Vector::Vector(const Vector& v)
23. {
24. size = v.size;
25. beg = new int[size];
26. for (int i = 0; i < size; i++) beg[i] = v.beg[i];
27. }
28. Vector::~Vector()
29. {
30. if (beg != 0) delete[] beg;
31. }
32. const Vector& Vector::operator = (const Vector& v)
33. {
34. if (this == &v) return \*this;
35. if (beg != 0) delete[] beg;
36. size = v.size;
37. beg = new int[size];
38. for (int i = 0; i < size; i++) beg[i] = v.beg[i];
39. return \*this;
40. }
41. ostream& operator << (ostream& out, const Vector& v)
42. {
43. if (v.size == 0) out << "Empty" << endl;
44. else
45. {
46. for (int i = 0; i < v.size; i++) out << v.beg[i] << " ";
47. out << endl;
48. }
49. return out;
50. }
51. istream& operator >> (istream& in, Vector& v)
52. {
53. for (int i = 0; i < v.size; i++)
54. { cout << ">"; in >> v.beg[i]; }
55. return in;
56. }
57. int Vector::operator [] (int i)
58. {
59. if (i < 0) throw error("index < 0");
60. if (i >= size) throw error("index > size");
61. return beg[i];
62. }
63. Vector Vector::operator + (int a)
64. {
65. if (size + 1 > MAX\_SIZE) throw error("Vector is full");
66. Vector temp(size + 1, beg);
67. temp.beg[size] = a;
68. size++;
69. return temp;
70. }
71. Vector Vector::operator -- ()
72. {
73. if (size == 0) throw error("Vector is empty");
74. if (size == 1)
75. {
76. size = 0;
77. delete[] beg;
78. beg = 0;
79. return \*this;
80. }
81. Vector temp(size, beg);
82. delete[] beg;
83. size--;
84. beg = new int[size];
85. for (int i = 0; i < size; i++) beg[i] = temp.beg[i];
86. return \*this;
87. }

**Файл main.cpp:**

1. #include "Vector.h"
2. #include "Error.h"
3. #include <iostream>
4. using namespace std;
6. int main()
7. {
8. try
9. {
10. Vector x(2);
11. Vector y;
12. cout << x;
13. cout << "Number? ";
14. int i; cin >> i;
15. cout << x[i] << endl;
16. y = x + 3;
17. cout << y;
18. --x;
19. cout << x;
20. --x;
21. cout << x;
22. --x;
23. }
24. catch (error e)
25. {
26. e.what();
27. }
29. return 0;
30. }

**Скриншоты результатов работы программы**

****

****

**Ответы на вопросы**

