|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 7** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Алгоритмические стратегии или методы разработки алгоритмов. Перебор и методы его сокращения.»** | |
|  | |
| Выполнил студент группы ИКБО-33-21 | Зарожина Я.А. |
| Принял преподаватель | Муравьева Е.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2022

# **Цель работы**

Получить навыки применения методов, позволяющих сократить число переборов в задачах, которые могут быть решены только методом перебора всех возможных вариантов решения.

# **Постановка задачи**

Разработать алгоритм решения задачи с применением метода, указанного в варианте и реализовать программу.

1) Оценить количество переборов при решении задачи стратегией «в лоб» - грубой силы

2) Привести анализ снижения числа переборов при применении метода.

3) Оформить отчет в соответствии с требованиями документирования разработки ПО: Постановка задачи, Описание алгоритмов и подхода к решению, Код, результаты тестирования.

Условие задания:

|  |  |  |
| --- | --- | --- |
| Номер варианта | Задача | Метод |
| 11 | Разработать программу расстановки на стандартной 64-клеточной шахматной доске 8 ферзей так, чтобы ни один из них не находился под боем другого. | метода ветвей и границ |

# **Решение**

Создадим две структуры. В первой структуре Point лежат координаты ферзей. Во второй структуре ferzes тоже лежат координаты ферзей, которые содержат примитивные конструкторы.

|  |
| --- |
| struct Point{  int x;  int y;  };  struct ferzes  {  int x, y;  ferzes(int m\_x = 0, int m\_y = 0)  {  x = m\_x;  y = m\_y;  }  }; |

Далее была создана функция заполнения поля нулями:

|  |
| --- |
| void Fill(int\*\* pole, int N)  {  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++) pole[i][j] = 0;  }  } |

Затем была создана функция для вывода поля в консоль:

|  |
| --- |
| void print(int\*\* pole, int N)  {  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++) cout << pole[i][j] << ' ';  cout << endl;  }  } |

А также была создана функция для вывода поля в блокнот:

|  |
| --- |
| void Pr\_In\_File(int\*\* pole, int N, ofstream\* FILE)  {  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++)  {  if(pole[i][j] == 0) \*FILE << "o ";  else \*FILE << "x ";  }  \*FILE << endl;  }  } |

Далее была создана функция проверки ферзей. Т.е. проверяем бьет ли наш ферзь, тех ферзей, которые уже есть на поле:

|  |
| --- |
| bool Is\_Kill(ferzes ferz, ferzes\* all\_ferz, int N)  {  for (int i = 0; i < N; i++)  {  int dx = abs(ferz.x - all\_ferz[i].x);  int dy = abs(ferz.y - all\_ferz[i].y);  if (dx == 0 || dy == 0 || (dx == dy)) return true;  }  return false;  } |

Затем в методе main пишем алгоритм работы программы для первого случая. Создаем массив из ферзей, вводим переменные, которые отвечают за расстояние между ферзями, а дальше работаем с циклом. Расставляем ферзей в таком порядке, чтобы ни один из них не находился под боем другого.

|  |
| --- |
| Point Ferz[num\_fer];  int dx,dy;  bool flag=true;  do  {  for(int i=0;i<num\_fer;i++)  {  cost=0;  while(flag)  {  if(i>=(num\_fer/2+1)) cost++;  if(cost==15)  {  i=num\_fer;  break;  }  Ferz[i].x=rand()%num\_fer+1;  Ferz[i].y=rand()%num\_fer+1;  if(i==0) flag=false;  for(int j=0;j<i;j++)  {  dx=abs(Ferz[i].x-Ferz[j].x);  dy=abs(Ferz[i].y-Ferz[j].y);  if(dx==dy || dy==0 || dx==0)  {  flag=true;  break;  }  else flag=false;  }  }  flag=true;  }  }while(cost==15);  //Расстановка ферзей  cout<<' '<<' ';  if(num\_fer>=10) cout<<' ';  bool on\_place;  for(int i=0;i<num\_fer;i++) cout<<i+1<<' ';  cout<<endl;  for(int i=0;i<num\_fer;i++)  {  cout<<i+1<<' ';  if(i<(9) && num\_fer>=10) cout<<' ';  for(int j=0;j<num\_fer;j++)  {  on\_place=false;  for(int k=0;k<num\_fer;k++)  {  if(Ferz[k].y==i+1 && Ferz[k].x==j+1)  {  cout<<'x'<<' ';  on\_place=true;  break;  }  }  if(!on\_place) cout<<'o'<<' ';  if(j>=9) cout<<' ';  }  cout<<endl;  }  } |

Во втором случае мы подсчитываем благоприятное количество решений с расстановкой ферзей.

|  |
| --- |
| cout << "Введите количество фрезей:";  cin >> N;  if (N == 0 || N == 2 || N == 3)  {  cout << "Подсчет решений:" << 0 << endl;  return 0;  }  else if (N == 1)  {  cout << "Подсчет решений:" << 1 << endl;  return 0;  }  ofstream FILE;  FILE.open("output.txt");  int solution = 0, count\_ferz = 0;  int\*\* field = new int\*[N];  for (int i = 0; i < N; i++) field[i] = new int[N];  ferzes\* all\_ferzes = new ferzes[N];  Fill(field, N);  int start = clock();  cout << "Начало решения..." << endl;  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++)  {  ferzes temp(j, i);  bool result = Is\_Kill(temp, all\_ferzes, count\_ferz); //false - поставить можно  if (!result)  {  field[i][j] = 1;  all\_ferzes[count\_ferz].x = j;  all\_ferzes[count\_ferz].y = i;  count\_ferz++;  if (count\_ferz == N)  {  FILE << solution + 1 << '.' << endl;  Pr\_In\_File(field, N, &FILE);  FILE << endl;  solution++;  field[all\_ferzes[count\_ferz - 1].y][all\_ferzes[count\_ferz - 1].x] = 0;  i = all\_ferzes[count\_ferz - 1].y;  j = all\_ferzes[count\_ferz - 1].x;  count\_ferz--;  if((j + 1) == N && count\_ferz != 0)  {  field[all\_ferzes[count\_ferz - 1].y][all\_ferzes[count\_ferz - 1].x] = 0;  i = all\_ferzes[count\_ferz - 1].y;  j = all\_ferzes[count\_ferz - 1].x;  count\_ferz--;  }  continue;  }  break;  }  else if ((j + 1) == N)  {  field[all\_ferzes[count\_ferz - 1].y][all\_ferzes[count\_ferz - 1].x] = 0;  i = all\_ferzes[count\_ferz - 1].y;  j = all\_ferzes[count\_ferz - 1].x;  count\_ferz--;  if ((j+1) == N && count\_ferz != 0)  {  field[all\_ferzes[count\_ferz - 1].y][all\_ferzes[count\_ferz - 1].x] = 0;  i = all\_ferzes[count\_ferz - 1].y;  j = all\_ferzes[count\_ferz - 1].x;  count\_ferz--;  }  }  }  }  int end = clock();  cout << "Подсчет решений:" << solution << endl;  cout << "Время решения:" << (end - start) / CLOCKS\_PER\_SEC << " seconds" << endl;  for (int i = 0; i < N; i++) delete field[i];  delete[] field;  delete[] all\_ferzes;  FILE.close();  system("notepad output.txt");  system("pause");  }  if(number==0){  break;  } |

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ.

При первом заходе программа выводит на экран свой основной интерфейс, где пользователю предлагается выбрать задание на рисунке 1:

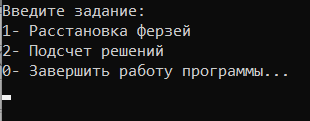


Рисунок 1 – Интерфейс программы

При выборе пункта 1 имеем вывод расстановки ферзей:

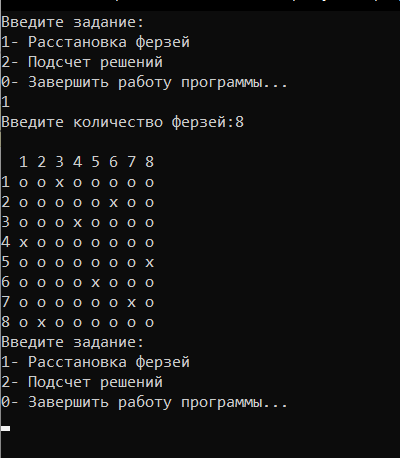


Рисунок 2 – Расстановка 8 ферзей на поле 8x8

При выборе пункта 2 имеем вывод подсчета решений и решения в открывшемся файле (Рисунок 3), (Рисунок 4), (Рисунок 5), (Рисунок 6), (Рисунок 7).

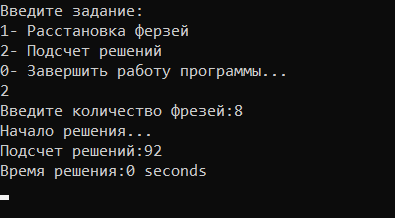


Рисунок 3 – Подсчет решений

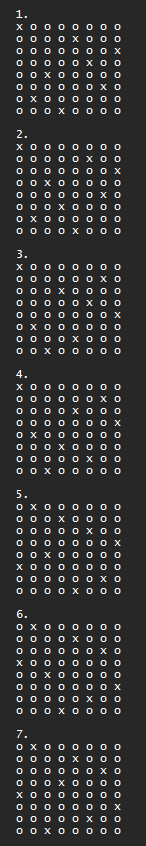
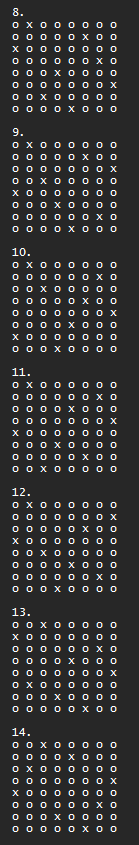
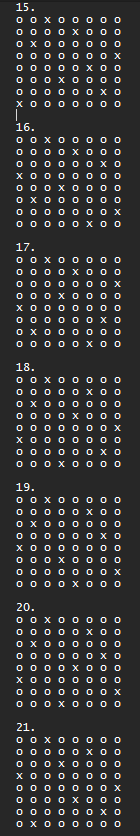
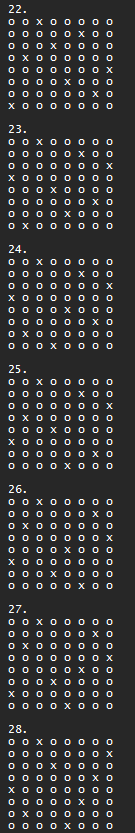
   

Рисунок 4 – Варианты решений из файла (1)

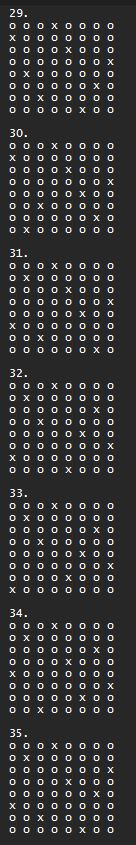
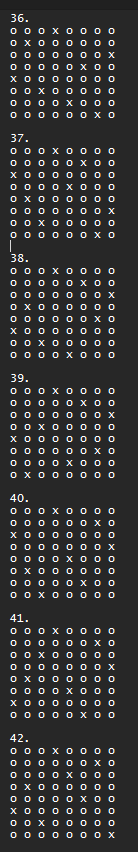
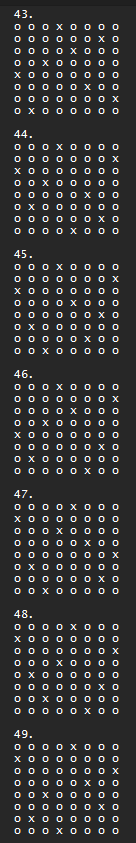
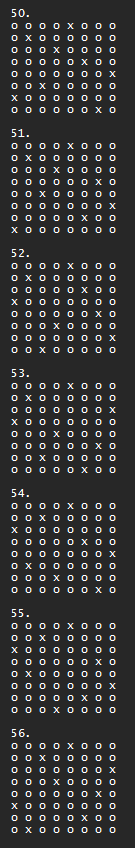
   

Рисунок 5 – Варианты решений из файла (2)

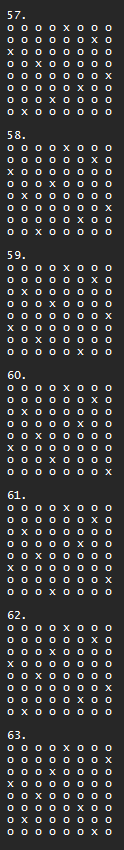
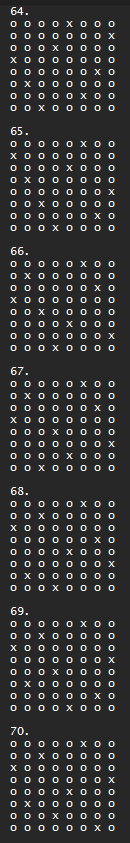
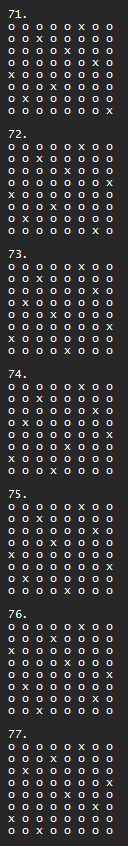
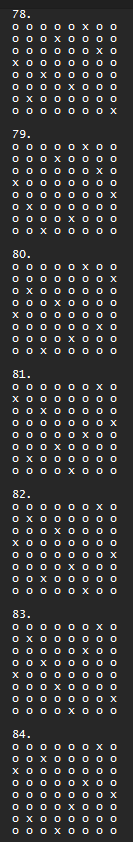
   

Рисунок 6 – Варианты решений из файла (3)

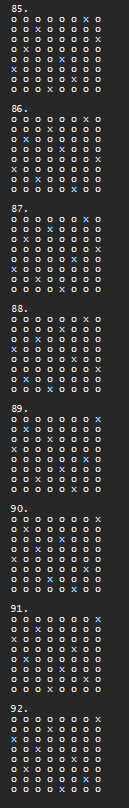


Рисунок 7 – Варианты решений из файла (4)

# **Вывод**

В результате выполнения работы я получила практические навыки по выполнению применения методов, позволяющих сократить число переборов в задачах, которые могут быть решены только методом перебора всех возможных вариантов решения.

# **Исходный код программы**

|  |
| --- |
| #include <iostream>  #include <math.h>  #include <locale>  #include <ctime>  #include <fstream>  using namespace std;  struct Point{  int x;  int y;  };  struct ferzes  {  int x, y;  ferzes(int m\_x = 0, int m\_y = 0)  {  x = m\_x;  y = m\_y;  }  };  void Fill(int\*\* pole, int N)  {  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++) pole[i][j] = 0;  }  }  void print(int\*\* pole, int N)  {  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++) cout << pole[i][j] << ' ';  cout << endl;  }  }  void Pr\_In\_File(int\*\* pole, int N, ofstream\* FILE)  {  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++)  {  if(pole[i][j] == 0) \*FILE << "o ";  else \*FILE << "x ";  }  \*FILE << endl;  }  }  bool Is\_Kill(ferzes ferz, ferzes\* all\_ferz, int N)  {  for (int i = 0; i < N; i++)  {  int dx = abs(ferz.x - all\_ferz[i].x);  int dy = abs(ferz.y - all\_ferz[i].y);  if (dx == 0 || dy == 0 || (dx == dy)) return true;  }  return false;  }  int main()  {  setlocale(LC\_ALL,"Russian");  int N;  int number=0;  while(true){  cout << "Введите задание: \n1- Расстановка ферзей\n2- Подсчет решений\n0- Завершить работу программы...\n";  int number=0;  cin >> number;  if(number==1)  {  srand(time(NULL));  unsigned num\_fer,cost;  cout<<"Введите количество ферзей:";  cin>>num\_fer;  while(num\_fer<4)  {  cout<<"Неверно. Введите еще раз:";  cin>>num\_fer;  }  cout<<endl;  Point Ferz[num\_fer];  int dx,dy;  bool flag=true;  do  {  for(int i=0;i<num\_fer;i++)  {  cost=0;  while(flag)  {  if(i>=(num\_fer/2+1)) cost++;  if(cost==15)  {  i=num\_fer;  break;  }  Ferz[i].x=rand()%num\_fer+1;  Ferz[i].y=rand()%num\_fer+1;  if(i==0) flag=false;  for(int j=0;j<i;j++)  {  dx=abs(Ferz[i].x-Ferz[j].x);  dy=abs(Ferz[i].y-Ferz[j].y);  if(dx==dy || dy==0 || dx==0)  {  flag=true;  break;  }  else flag=false;  }  }  flag=true;  }  }while(cost==15);  //Расстановка ферзей  cout<<' '<<' ';  if(num\_fer>=10) cout<<' ';  bool on\_place;  for(int i=0;i<num\_fer;i++) cout<<i+1<<' ';  cout<<endl;  for(int i=0;i<num\_fer;i++)  {  cout<<i+1<<' ';  if(i<(9) && num\_fer>=10) cout<<' ';  for(int j=0;j<num\_fer;j++)  {  on\_place=false;  for(int k=0;k<num\_fer;k++)  {  if(Ferz[k].y==i+1 && Ferz[k].x==j+1)  {  cout<<'x'<<' ';  on\_place=true;  break;  }  }  if(!on\_place) cout<<'o'<<' ';  if(j>=9) cout<<' ';  }  cout<<endl;  }  }  if(number==2)  {  cout << "Введите количество фрезей:";  cin >> N;  if (N == 0 || N == 2 || N == 3)  {  cout << "Подсчет решений:" << 0 << endl;  return 0;  }  else if (N == 1)  {  cout << "Подсчет решений:" << 1 << endl;  return 0;  }  ofstream FILE;  FILE.open("output.txt");  int solution = 0, count\_ferz = 0;  int\*\* field = new int\*[N];  for (int i = 0; i < N; i++) field[i] = new int[N];  ferzes\* all\_ferzes = new ferzes[N];  Fill(field, N);  int start = clock();  cout << "Начало решения..." << endl;  for (int i = 0; i < N; i++)  {  for (int j = 0; j < N; j++)  {  ferzes temp(j, i);  bool result = Is\_Kill(temp, all\_ferzes, count\_ferz); //false - поставить можно  if (!result)  {  field[i][j] = 1;  all\_ferzes[count\_ferz].x = j;  all\_ferzes[count\_ferz].y = i;  count\_ferz++;  if (count\_ferz == N)  {  FILE << solution + 1 << '.' << endl;  Pr\_In\_File(field, N, &FILE);  FILE << endl;  solution++;  field[all\_ferzes[count\_ferz - 1].y][all\_ferzes[count\_ferz - 1].x] = 0;  i = all\_ferzes[count\_ferz - 1].y;  j = all\_ferzes[count\_ferz - 1].x;  count\_ferz--;  if((j + 1) == N && count\_ferz != 0)  {  field[all\_ferzes[count\_ferz - 1].y][all\_ferzes[count\_ferz - 1].x] = 0;  i = all\_ferzes[count\_ferz - 1].y;  j = all\_ferzes[count\_ferz - 1].x;  count\_ferz--;  }  continue;  }  break;  }  else if ((j + 1) == N)  {  field[all\_ferzes[count\_ferz - 1].y][all\_ferzes[count\_ferz - 1].x] = 0;  i = all\_ferzes[count\_ferz - 1].y;  j = all\_ferzes[count\_ferz - 1].x;  count\_ferz--;  if ((j+1) == N && count\_ferz != 0)  {  field[all\_ferzes[count\_ferz - 1].y][all\_ferzes[count\_ferz - 1].x] = 0;  i = all\_ferzes[count\_ferz - 1].y;  j = all\_ferzes[count\_ferz - 1].x;  count\_ferz--;  }  }  }  }  int end = clock();  cout << "Подсчет решений:" << solution << endl;  cout << "Время решения:" << (end - start) / CLOCKS\_PER\_SEC << " seconds" << endl;  for (int i = 0; i < N; i++) delete field[i];  delete[] field;  delete[] all\_ferzes;  FILE.close();  system("notepad output.txt");  system("pause");  }  if(number==0){  break;  }  }  return 0;  } |