Освобождение ресурсов

Необходимо помнить, что любое выделение динамической памяти с помощью оператора `new` (без использования `std::nothrow`) потенциально может выбросить исключение и привести к завершению программы. При этом нужно убедиться, что все ранее выделенные ресурсы освобождены.

Проблема:

Предположим, понадобилось выделить память для трёх строк (или других массивов)

```
char* a = new char [10];
char* b = new char [10]; // возможно std::bad_alloc => утечка памяти
char* c = new char [10]; // то же самое
// какие-то действия
// ...
delete[] a; // до сюда можем не дойти
delete[] b;
delete[] c;
```

Возможное решение:

```
char* a = nullptr;
char* b = nullptr;
char* c = nullptr;
try
{
  a = new char[10];
  b = new char[10];
  c = new char[10];
  // какие-то действия
  // ...
  delete[] a;
  delete[] b;
  delete[] c;
}
catch (...)
{
  delete[] a;
  delete[] b;
  delete[] c;
}
```

Ещё один вариант:

В некоторых случаях, если это удобно, можно выделить память только один раз и разместить в ней все строки сразу

```
char* a = new char[10 * 3];
char* b = a + 10;
char* c = a + 20;
// какие-то действия
// ...
delete[] a;
```

Для двумерного динамического массива:

```
int nRow = 10;
int nCol = 10;
int** matrix = new int*[nRow];
for (int i = 0; i < nRow; ++i)
{
  matrix[i] = new int[nCol];  // возможно std::bad_alloc => утечка памяти
}
// какие-то действия
// ...
for (int i = 0; i < nRow; ++i)
{
  delete[] matrix[i]; // до сюда можем не дойти
}
delete[] matrix;
```

Возможное решение:

```
int nRow = 10;
int nCol = 10;
int** matrix = nullptr;
try
{
  matrix = new int* [nRow] {nullptr}; // инициализация C++11
  for (int i = 0; i < nRow; ++i)
  {
    matrix[i] = new int[nCol];
  }
  // какие-то действия
  // ...
}
catch (...)
{
  // обработка ошибки
}
if (matrix)
{
  for (int i = 0; i < nRow; ++i)
  {
    delete[] matrix[i];
  }
  delete[] matrix;
}
```

Выделение и освобождение памяти можно вынести в отдельные функции:

```
void allocateMatrix(int**& matrix, int nRow, int nCol)
{
  matrix = new int* [nRow] {nullptr}; // инициализация C++11
  for (int i = 0; i < nRow; ++i)
  {
    matrix[i] = new int[nCol];
  }
}
```

```
void deallocateMatrix(int** matrix, int nRow)
{
  if (matrix)
  {
    for (int i = 0; i < nRow; ++i)
    {
      delete[] matrix[i];
    }
    delete[] matrix;
  }
}
```

Тогда:

```
int nRow = 10;
int nCol = 10;
int** matrix = nullptr;
try
{
  allocateMatrix(matrix, nRow, nCol);
  // какие-то действия
  // ...
}
catch (...)
{
  // обработка ошибки
}
deallocateMatrix(matrix, nRow);
```

Ещё один вариант:

Можно сэмулировать работу с двумерным динамическим массивом через одномерный

```
int nRow = 10;
int nCol = 10;
int* matrix = new int[nRow * nCol];
// какие-то действия
// вместо matrix[i][j] используем matrix[i * nCol + j]
delete[] matrix;
```