**Лабораторная работа №3**

Тема: Разработка приложения Сапер.

Цель: освоить на практике использование матриц. Закрепить знания о создании приложений в C++Builder.

Ход выполнения лабораторной работы

1. Решить задачи указанные в приложении А;
2. Составить отчет;
3. Предоставить листинг кода программ;
4. Написать комментарий для каждой строчки кода;
5. Предоставить скриншоты выполнения программ.

Приложение А – задачи для лабораторной работы №1

# Часть 1. Размещаем бомбы и числа

Размещаем на форме таблицу строк **StringGrid** и кнопку. Устанавливаем ширину и высоту строк по умолчанию 20.

Для начала количество строк и колонок устанавливаем 9.

Для игры нам понадобится двумерный массив **A,** в котором будет храниться поле с бомбами и цифрами, этот массив нам ну- жен, т.к. в таблицу строк на форму выводится только часть этого поля, по мере его раскрывания игроком.

В процедуре **FormCreate** осуществляем заполнение масси- ва **A** в два этапа:

1. Обнуление массива.
2. Заполнение бомбами.
3. Расстановка чисел вокруг бомб (каждое число показыва- ет, сколько бомб находится в соседних с этим числом ячейках)

Подробнее:

1. Заполняем массив нулями.
2. Бомбы (10 штук) ставятся случайным образом. (Если ваш двумерный массив **A** состоит из целых чисел **integ- er,** то для хранения бомбы предлагается использовать число **-1**, для хранения пустого поля - число **0**)
3. Алгоритм расстановки чисел, окружающих бомбы сле- дующий:

Каждый элемент массива, уже заполненного бомбами (числами **-1**) и пустыми полями (нолями), просматривается один за другим и если найдена бомба, то к каждой ячейке, которая находится по соседству с бомбой, прибавляем единицу (прибавляем к тому, что уже содержала ячейка).

Допустим вот участок массива и две бомбы:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |  | 1 | 1 | 1 | 0 | 0 |  | 1 | 1 | 1 | 0 | 0 |
| 0 | **-1** | 0 | 0 | 0 |  | 1 | **-1** | 1 | 0 | 0 |  | 1 | **-1** | 2 | 1 | 1 |
| 0 | 0 | 0 | **-1** | 0 |  | 1 | 1 | 1 | **-1** | 0 |  | 1 | 1 | 2 | **-1** | 1 |
| 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 1 | 1 | 1 |

После заполнения массива **A** выводим его в таблицу строк **StringGrid,** при этом если в ячейке бомба – то выводим букву **Б**, если пустое поле, т.е. ноль – ничего не выводим.

В результате завершения **части 1**, у вас по нажатию на кнопку **REBOOT** должна появляться примерно такая картина, какая показана на рисунке выше.

# Примечания:

1. При расстановке бомб следите, чтобы несколько бомб не «расставились» в одну ячейку – т.е. в результате должно быть расставлено ровно 10 бомб и все в разных ячейках.
2. При окружении «бомб» числами следите, чтобы если в соседней с бомбой ячейке находится другая бомба, то число туда, естественно, прибавляться не должно.

# Часть 2. Определение координаты щелчка мышью

Для игры требуется знать, в какую ячейку таблицы произошел щелчок мыши. Создаем процедуру **MouseUp** для таблицы строк**.** В ней используем процедуру

***SG.MouseToCell(X,Y,X1,Y1);***

*Где SG – таблица строк*

*X, Y – координаты щелчка мыши на таблице строк*

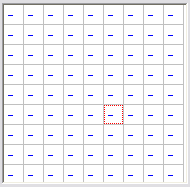
*X1, Y1 – координаты ячейки в таблице (X1 – столбец, Y1 - строка)*

Эта процедура переводит координаты щелчка в табличные координаты ячейки, в которую был произведен щелчок.

# Задание:

Сделайте так, чтобы в метку над таблицей выводились коор- динаты ячейки, в которую щелкнули мышью (рис. левее)

# Часть 3. Начинаем игру



Теперь при запуске приложения, поле выглядит, как показано на рис. выше. Т.е. содержание мас- сива **A** неизвестно.

Щелкая по ячейкам таблицы – они должны раскрываться и в зависимости от своего содержимого должны происходить различные события (Все что должно происходить можно пронаблюдать в игре **Сапер** от MS Windows):

1. Если в ячейке бомба – раскрывается местонахождение всех остальных бомб и игра заканчивается
2. Если в ячейке цифра – то эта цифра просто раскрывается.
3. Если в ячейке пустое поле (т.е. ноль) – то раскрываются все остальные пустые поля до тех пор, пока не откроются цифры (достаточно сложно объяснить, поэтому смотрим реальную игру)