Попробовал подробно расписать архитектуру:

Имеющиеся классы:

1. game:

Поля:

field - само поле

(матрица [высота][ширина])

variants - таблица с вариантами.

(матрица [высота][ширина])

Имеющиеся функции:

1. Функция подбирает для заданной точки i все возможные варианты, заносит все найденные варианты в соответствующее место таблицы вариантов variants[i] и возвращает количество найденных вариантов. Прототип ):

int getVariants(field, variants, i)

где field - матрица с текущим состоянием кроссворда, на данном этапе решения, variants - таблица вариантов, i - номер элемента. Можно использовать номер столбца и строки, но я обычно использую один номер и обращение к элементу делаю так: field[int(i / \*высота\*)][i % \*ширина\*], тут, как тебе будет удобнее, я подстроюсь :)

2. Функция, которая, собственно, решает кроссворд, заполняет варианты. Возвращает решено верно или не верно. Прототип:

bool solveCrossword(crossword)

где crossword - объект класса game.

Алгоритм:

1. Читаем входной файл.

2. Запускаем рекурсию функцией solveCrossword, в ней:

2.1 Считаем вариативность для всех элементов поля, функцией getVariants.

2.2 Для всех элементов поля с вариативность 1 заполняем решение.

2.3 Если есть элементы с вариативностью 0, значит, мы ошиблись с комбинацией и надо вернуть рекурсию на уровень выше - return False

2.4 Если был хотя бы один элемент с вариативностью 1, его заполнения могло всё поменять и надо вернуться к пункту 2.1, если таких элементов не было, идём к пункту 2.5.

2.5 Поскольку сколько то элементов (с вариантивность 1) мы уже заполнили, проверим, есть ли неизвестные элементы, а то может всё решено. Если всё решено - вышли с True.

2.6 Берём элемент с наименьшей вариативностью (в большинстве случаев - 2), делаем предположение (подставляем один из вариантов) и вызываем себя рекурсией. Если функция вызванная рекурсивно сказала, что с нашим предположением решений нет (вернула False) делаем другое предположение и вызываем её снова. Если варианты кончились, а решений нет, значит, мы ошиблись где то до этого, сами возвращаем False и вернём рекурсию на уровень выше.

2.7 Если мы тут, значит все уровени рекурсии ниже нашего вернули по цепочки True, значит, наше предположение верно, заполняем решение для этой клетки и возвращаем True.

3. Выводим в выходной файл, потом подумаем о красивой рисовалке результата :)

Мысли: как уже говорил, вся магия и все непонятки в функции getVariants... Насколько я понял из разговоров на занятиях, народ пишет "эвристики" отдельные на каждый вариант, т.е. если длина пути должна быть 2, то берём такие то варианты, если 3 то такие то. Как сказал Иван, 150 if-ов :)

Т.к. для получения зачёта наша программа должна решать 3 примера, которые лежат в Wiki, а там максимальная длина пути 8, есть смысл расписывать такие "эвристики" только до 8...

Ну или как то иначе это писать...