Lösungsvorschläge zu den Karel Aufgaben

0.0.1 karelsFirstProgram

```
void karelsFirstProgram() {
    moveForward();
    pickBeeper();
    moveForward();
    turnLeft();
    moveForward();
    turnRight();
    moveForward();
    dropBeeper();
    moveForward();
}
```

1.1.1 obtainArtefact

```
void obtainArtifact() {
    turnRight();
    moveForward();
    turnLeft();
    repeat (3) {
        moveForward();
    }
    turnLeft();
    moveForward();
    pickBeeper();
    moveForward();
    turnLeft();
    repeat (3) {
        moveForward();
    }
    turnLeft();
    moveForward();
    dropBeeper();
}
```

1.1.2 defuseOneBomb

```
void defuseOneBomb() {
    repeat (9) {
        moveForward();
    }
    pickBeeper();
    turnAround();
    repeat (9) {
        moveForward();
    }
    turnAround();
}
```

1.1.3 defuseTwoBomb

```
void defuseTwoBombs() {
    defuseOneBomb();
    turnLeft();
    defuseOneBomb();
}
```

1.1.4 practiceHomeRun

```
void practiceHomeRun() {
    repeat (4) {
        repeat (9) {
            moveForward();
        }
        pickBeeper();
        turnLeft();
    }
}
```

1.2.1 climbTheStairs

```
void climbTheStairs() {
    moveForward();
    repeat (6) {
        turnLeft();
        moveForward();
        turnRight();
        moveForward();
    }
}
```

1.2.2 fillTheHoles

```
void fillTheHoles() {
    repeat (4) {
        moveForward();
        turnRight();
        moveForward();
        dropBeeper();
        turnAround();
        moveForward();
        turnRight();
        moveForward();
    }
}
```

1.2.3 saveTheFlower

```
void saveTheFlower() {
    moveForward();
    pickBeeper();
    repeat (4) {
        turnLeft();
        moveForward();
        moveForward();
        turnRight();
        moveForward();
    }
    dropBeeper();
    repeat (4) {
        moveForward();
        turnRight();
        moveForward();
        moveForward();
        turnLeft();
    }
}
```

1.2.4 mowTheLawn

```
void mowTheLawn() {
    repeat (2) {
        mow();
        turnUpLeft();
        mow();
        turnUpRight();
    }
    mow();
    turnUpLeft();
    mow();
}

void mow() {
    repeat (6) {
        moveForward();
        pickBeeper();
    }
    moveForward();
}

void turnUpLeft() {
    turnLeft();
    moveForward();
    turnLeft();
}

void turnUpRight() {
    turnRight();
    moveForward();
    turnRight();
}
```

1.3.1 harvestTheField

```
void harvestTheField() {
    harvestHalf();

    turnRight();
    moveForward();
    turnRight();
    moveForward();

    harvestHalf();
}

void harvestHalf() {
    harvestLine();

    moveForward();
    turnLeft();
    moveForward();
    turnLeft();

    harvestLine();
}

void harvestLine() {
    repeat (3) {
        pickBeeper();
        turnRight();
        moveForward();
        turnLeft();
        moveForward();
    }
    pickBeeper();
}
```

1.3.2 repairTheStreet

```
void repairTheStreet() {
    repeat (9) {
        maybeRepair();
        moveForward();
    }
    maybeRepair();
}

void maybeRepair() {
    if (rightIsClear()) {
        turnRight();
        moveForward();
        dropBeeper();
        turnAround();
        moveForward();
        turnRight();
    }
}
```

1.3.3 cleanTheRoom

```
void cleanTheRoom() {
    repeat (5) {
        cleanLine();
        turnUpLeft();
        cleanLine();
        if (rightIsClear()) {
            turnUpRight();
        }
    }
}

void cleanLine() {
    repeat (10) {
        if (onBeeper()) {
            pickBeeper();
        }
        if (frontIsClear()) {
            moveForward();
        }
    }
}

void turnUpLeft() {
    turnLeft();
    moveForward();
    turnLeft();
}

void turnUpRight() {
    turnRight();
    moveForward();
    turnRight();
}
```

1.3.4 tileTheFloor

```
void tileTheFloor() {
    repeat (100) {
        dropBeeper();
        if (!frontIsClear() || beeperAhead()) {
            turnLeft();
        }
        moveForward();
    }
}
```

1.4.1 stealOlympicFire

```
void stealOlympicFire() {
    moveForward();
    repeat (6) {
        turnLeft();
        moveForward();
        turnRight();
        moveForward();
    }
    pickBeeper();
    moveForward();
    turnRight();
    repeat (6) {
        moveForward();
    }
    turnLeft();
    moveForward();
}
```

1.4.2 removeTheTiles

```
void removeTheTiles() {
    repeat (100) {
        pickBeeper();
        if (!frontIsClear() || !beeperAhead()) {
            turnLeft();
        }
        moveForward();
    }
}
```

1.4.3 walkTheLabyrinth

```
void walkTheLabyrinth() {
    repeat (99) {
        if (!frontIsClear()) {
            if (leftIsClear()) {
                turnLeft();
            } else {
                turnRight();
            }
        }
        moveForward();
    }
}
```

2.1.1 hangTheLampions

```
void hangTheLampions() {
    while (onBeeper()) {
        turnLeft();
        pickBeeper();
        while (frontIsClear()) {
            moveForward();
        }
        dropBeeper();
        turnAround();
        while (frontIsClear()) {
            moveForward();
        }
        turnLeft();
        if (frontIsClear()) {
            moveForward();
        }
    }
}
```

2.1.2 followTheSeeds

```
void followTheSeeds() {
    while (beeperAhead()) {
        moveForward();
        pickBeeper();
        if (!beeperAhead()) {
            turnLeft();
        }
    }
}
```

2.1.3 cleanTheTunnels

```
void cleanTheTunnels() {
    repeat (10) {
        if (onBeeper()) {
            turnLeft();
            pickBeeper();
            while (beeperAhead()) {
                moveForward();
                pickBeeper();
            }
            turnAround();
            while (frontIsClear()) {
                moveForward();
            }
            turnLeft();
        }
        if (frontIsClear()) {
            moveForward();
        }
    }
}
```

2.2.1 increment

```
void increment() {
    while (onBeeper()) {
        pickBeeper();
        moveForward();
    }
    dropBeeper();
}
```

2.2.2 decrement

```
void decrement() {
    while (!onBeeper()) {
        dropBeeper();
        if (frontIsClear()) {
            moveForward();
        }
    }
    pickBeeper();
}
```

2.2.3 addSlow

Notiz:
increment() kann aus 2.2.1 wiederverwendet werden.
decrement() kann aus 2.2.2 wiederverwendet werden.

```
void addSlow() {
    decrement();
    if (frontIsClear()) {
        walkBack();
        increment();
        walkBack();
        addSlow();
    }
}

void walkBack() {
    turnAround();
    while (frontIsClear()) {
        moveForward();
    }
    if (!leftIsClear()) {
        turnRight();
        moveForward();
        turnRight();
    } else {
        turnLeft();
        moveForward();
        turnLeft();
    }
}
```

2.3.1 saveTheFlowers

```
void saveTheFlowers() {
    if (leftIsClear()) {
        turnLeft();
        while (!rightIsClear()) {
            moveForward();
        }
        turnRight();
        moveForward();
        if (onBeeper()) {
            pickBeeper();
            saveTheFlowers();

            dropBeeper();
            moveForward();
            turnRight();
            while (frontIsClear()) {
                moveForward();
            }
            turnLeft();
        }

    }
}
```

2.3.2 findTeddyBear

```
void findTeddyBear() {
    while (!onBeeper()) {
        if (!frontIsClear() && leftIsClear()) {
            turnLeft();
        } else if (!frontIsClear()) {
            turnAround();
        }
        moveForward();
    }
}
```

2.3.3 jumpTheHurdles

```
void jumpTheHurdles() {
    while (!onBeeper()) {
        turnLeft();
        while (!rightIsClear()) {
            moveForward();
        }
        turnRight();
        moveForward();
        turnRight();
        while (frontIsClear()) {
            moveForward();
        }
        turnLeft();
    }
}
```

2.4.1 solveTheMaze

```
void solveTheMaze() {
    while (!onBeeper()) {
        if (leftIsClear()) {
            turnLeft();
            moveForward();
        } else if (frontIsClear()) {
            moveForward();
        } else if (rightIsClear()) {
            turnRight();
            moveForward();
        } else {
            turnAround();
            moveForward();
        }
    }
}
```

2.4.2 quantizeBits

```
void quantizeBits() {
    repeat (10) {
        if (onBeeper()) {
            updateCode();
        }
        if (frontIsClear()) {
            moveForward();
        }
    }
}

void updateCode() {
    turnLeft();
    repeat (5) {
        moveForward();
    }
    if (onBeeper()) {
        while (frontIsClear()) {
            moveForward();
            if (!onBeeper()) {
                dropBeeper();
            }
        }
        turnAround();
        while (frontIsClear()) {
            moveForward();
        }
    } else {
        turnAround();
        repeat (5) {
            moveForward();
            if (onBeeper()) {
                pickBeeper();
            }
        }
```

```
    }
    turnLeft();
}
```

2.4.3 addFast

```
void addFast() {
    repeat (8) {
        if (onBeeper()) {
            moveForward();
            if (onBeeper() && beeperAhead()) {
                // 1 + 1 + 1
                moveForward();
                moveForward();
                dropBeeper();
                nextWithCarry();
            } else if (!onBeeper() && !beeperAhead()) {
                // 1 + 0 + 0
                moveForward();
                moveForward();
                dropBeeper();
                nextWithoutCarry();
            } else {
                // 1 + 1 + 0 OR 1 + 0 + 1
                moveForward();
                moveForward();
                nextWithCarry();
            }
        } else {
            moveForward();
            if (onBeeper() && beeperAhead()) {
                // 1 + 1
                moveForward();
                moveForward();
                nextWithCarry();
            } else if (!onBeeper() && !beeperAhead()) {
                // 0 + 0
                moveForward();
                moveForward();
                nextWithoutCarry();
            } else {
                // 1 + 0 OR 0 + 1
                moveForward();
                moveForward();
                dropBeeper();
                nextWithoutCarry();
            }
        }

        moveForward();
        moveForward();
        turnAround();
    }
}

void nextWithoutCarry() {
```

```
    turnRight();
    moveForward();
    turnRight();
    moveForward();
}

void nextWithCarry() {
    turnRight();
    moveForward();
    turnRight();
    moveForward();
    dropBeeper();
}
```

3.1.1 partyAgain

```
void partyAgain() {
    repeat (10) {
        turnLeft();
        pickBeeper();
        walkAndDrop();
        turnLeft();
        if (frontIsClear()) {
            moveForward();
        }
    }
}

void walkAndDrop() {
    if (frontIsClear()) {
        moveForward();
        walkAndDrop();
        moveForward();
    } else {
        dropBeeper();
        turnAround();
    }
}
```

3.2.1 secureTheCave

```
void secureTheCave() {
    repeat (10) {
        turnLeft();
        walk();
        pickAndDrop();
        turnRight();
        if (frontIsClear()) {
            moveForward();
        }
    }
}

void walk() {
    while (frontIsClear()) {
        moveForward();
    }
    turnAround();
}

void pickAndDrop() {
    if (onBeeper()) {
        pickBeeper();
        moveForward();
        pickAndDrop();
        dropBeeper();
        moveForward();
    } else {
        walk();
    }
}
```

3.2.2 layAndRemoveTiles

```
void layAndRemoveTiles() {
    if (onBeeper()) {
        turnAround();
    } else {
        dropBeeper();
        if (frontIsClear() && !beeperAhead()) {
            moveForward();
            layAndRemoveTiles();
            moveForward();
        } else {
            turnLeft();
            moveForward();
            layAndRemoveTiles();
            moveForward();
            turnRight();
        }
        pickBeeper();
    }
}
```

3.3.1 findShelters

```
void findShelters() {
    repeat (4) {
        if (frontIsClear() && !beeperAhead()) {
            moveForward();
            if (leftIsClear() || frontIsClear() || rightIsClear()) {
                dropBeeper();
                findShelters();
            }
            turnAround();
            moveForward();
            turnAround();
        }
        turnLeft();
    }
}
```

3.3.2 addSmart

```
void addSmart() {
    if (rightIsClear()) {
        if (onBeeper() && beeperAhead()) {
            // 1 + 1
            turnRight();
            moveForward();
            turnLeft();
            addSmartCarry();
        } else if (!onBeeper() && !beeperAhead()) {
            // 0 + 0
            turnRight();
            moveForward();
            turnLeft();
            addSmart();
        } else {
            // 1 + 0 OR 0 + 1
            addSumBit();
            addSmart();
        }
    }
}

void addSmartCarry() {
    if (rightIsClear()) {
        if (onBeeper() && beeperAhead()) {
            // 1 + 1 + 1
            addSumBit();
            addSmartCarry();
        } else if (!onBeeper() && !beeperAhead()) {
            // 1 + 0 + 0
            addSumBit();
            addSmart();
        } else {
            // 1 + 1 + 0 OR 1 + 0 + 1
            turnRight();
            moveForward();
            turnLeft();
            addSmartCarry();
        }
    }
}

void addSumBit() {
    moveForward();
    moveForward();
    dropBeeper();
    turnRight();
    moveForward();
    turnRight();
    moveForward();
    moveForward();
    turnAround();
}
```

3.3.3 computeFibonacci

```
void computeFibonacci() {
    repeat (8) {
        addSmart();
        moveForward();
        turnLeft();
        while (frontIsClear()) {
            moveForward();
        }
        turnRight();
    }
}
```