1 ascii2int

Lösungsidee:
Zuerst werden die Übergabeparamter überprüft. Wenn alle valid sind, wird zuerst eine integer (number) Variable deklariert und dann mit einer Schleife über das character array drüber iteriert und mit ‚switch case' bei jeder Dezimalziffer die entsprechende Nummer in eine Variable gespeichert, die dann anschließend mit 10 multiplieziert und dann zur Variable ‚number' addiert wird. Falls in der Schleife einen nicht dezimalen Character erreicht wird, wird aus der Schleife gesprungen.

Testfälle: „213532", „a213532", „213532-", „213 532" und „"

C-Program code:

Int main:

```c
int main(int argc, char* argv[]) {
    int test_ascii2int = 1;
    int test_triangularMatrix = 0;
    int test_interestBill = 0;
    int test_arithmeticAverage = 0;

    if (test_ascii2int) {
        printf("\n\n#### ascii2int ########################################################################\n\n");

        // test 1 ----------------
        ascii2int("213532");
        // test 2 ----------------
        ascii2int("a213532");
        // test 3 ----------------
        ascii2int("213532-");
        // test 4 ----------------
        ascii2int("213 532");
        // test 5 ----------------
        ascii2int("");
    }

    if (test_triangularMatrix) { ... }

    if (test_interestBill) { ... }

    if (test_arithmeticAverage) { ... }


    // just to make it more beautiful ----------------------------------------------------------------------------------
    printf("\n#############################################################################################\n\n");
    // -----------------------------------------------------------------------------------------------------------------

    return 0;
}
```

Int ascii2int:

```c
int ascii2int(char* chain) {
    printf("\nstring: \"%s\"", chain);

    if (chain[0] != '\0') {
        int number = 0;

        printf("\n\nconverting string to integer...");
        for (int i = 0; chain[i] != '\0'; i++) {
            int currNum = 0;

            switch (chain[i]) {
            case '0':
                currNum = 0;
                break;
            case '1':
                currNum = 1;
                break;
            case '2':
                currNum = 2;
                break;
            case '3':
                currNum = 3;
                break;
            case '4':
                currNum = 4;
                break;
            case '5':
                currNum = 5;
                break;
            case '6':
                currNum = 6;
                break;
            case '7':
                currNum = 7;
                break;
            case '8':
                currNum = 8;
                break;
            case '9':
                currNum = 9;
                break;

            default:
                currNum = -1;
                break;
            }

            // break if number is invalid (no decimal digit)
            if (currNum < 0) {
                break;
            }

            number *= 10;
            number += currNum;
        }
        printf("\nconverting finished\n");

        printf("\ninteger: %d", number);
    }
    else {
        printf("\nstring is empty!");
    }

    printf("\n************************************************************");

    return 0;
}
```

Test Ausgaben:
(Start values -> „213532“, „a213532“, „213532-“, „213 532“ und „“)

```
#### ascii2int ###############################################################################


string: "213532"

converting string to integer...
converting finished

integer: 213532
*********************************************************************
string: "a213532"

converting string to integer...
converting finished

integer: 0
*********************************************************************
string: "213532-"

converting string to integer...
converting finished

integer: 213532
*********************************************************************
string: "213 532"

converting string to integer...
converting finished

integer: 213
*********************************************************************
string: ""
string is empty!
*********************************************************************
#############################################################################################
```

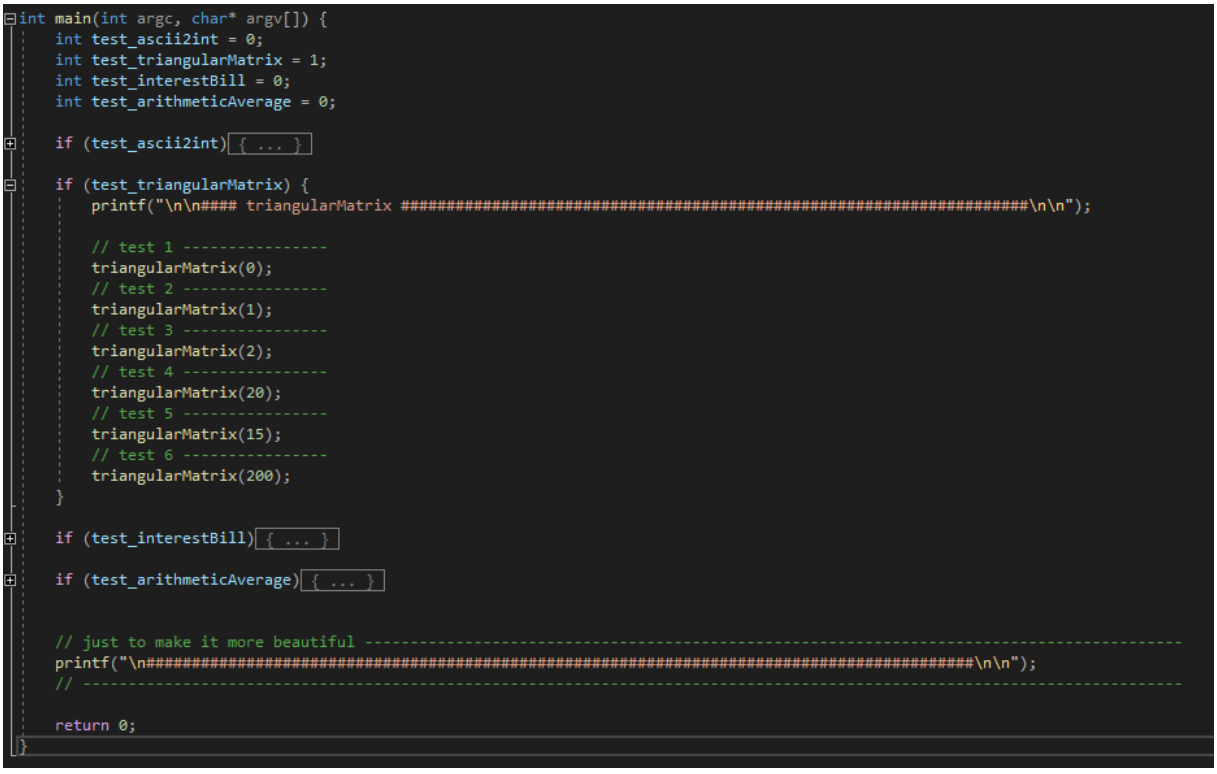
2 Dreiecksmatrix

Lösungsidee:
Zuerts werden die Übergabeparameter überprüft. Wenn alle valid sind wird in der Schleife jeder ungerade Wert wie eine linksbündige Pyramide ausgeben. Dabei fängt die Pyramide mit einem Wert an und wird bei jeder Stufe um eins erhöt. Um dies zu erreichen werden drei Variabeln benötigt. Eine die den ‚line break' bestimmt, eine die die ausgegebenen Nummern in der aktuellen Reihe zählt und eine die die ungerade Nummer besitzt. Wenn die ‚nummer zähler' Variable modulo mit der ‚line break' Variable gerechnet 0 ergibt wird ein ‚Line break' ausgeführt.

Testfälle: 0, 1, 2, 20, 15, 200

C-Program code:

Int main:



```c
int main(int argc, char* argv[]) {
    int test_ascii2int = 0;
    int test_triangularMatrix = 1;
    int test_interestBill = 0;
    int test_arithmeticAverage = 0;

    if (test_ascii2int) { ... }

    if (test_triangularMatrix) {
        printf("\n\n#### triangularMatrix ##########################################################################\n\n");

        // test 1 ----------------
        triangularMatrix(0);
        // test 2 ----------------
        triangularMatrix(1);
        // test 3 ----------------
        triangularMatrix(2);
        // test 4 ----------------
        triangularMatrix(20);
        // test 5 ----------------
        triangularMatrix(15);
        // test 6 ----------------
        triangularMatrix(200);
    }

    if (test_interestBill) { ... }

    if (test_arithmeticAverage) { ... }


    // just to make it more beautiful ------------------------------------------------------------------------------------------
    printf("\n###############################################################################################\n\n");
    // ------------------------------------------------------------------------------------------------------------------------

    return 0;
}
```

Int triangularMatrix:

```c
int triangularMatrix(int number) {
    printf("\nnumber = %d\n", number);

    if (number > 0) {
        int test_while = 1;
        int test_doWhile = 1;
        int test_for = 1;

        if (test_while) {
            printf("\nTest while\n\n");

            int lineBreakNum = 1;
            int printedNumCounter = 0;
            int i = 1;
            while (i < number) {
                printf("%d\t", i);
                printedNumCounter++;

                if ((printedNumCounter % lineBreakNum) == 0) {
                    printf("\n");
                    lineBreakNum++;
                    printedNumCounter = 0;
                }

                i += 2;
            }

            printf("\n\n");
        }

        if (test_doWhile) {
            printf("\nTest doWhile\n\n");

            int lineBreakNum = 1;
            int printedNumCounter = 0;
            int i = 1;

            if (i < number) {
                do {
                    printf("%d\t", i);
                    printedNumCounter++;

                    if ((printedNumCounter % lineBreakNum) == 0) {
                        printf("\n");
                        lineBreakNum++;
                        printedNumCounter = 0;
                    }

                    i += 2;
                } while (i < number);
            }

            printf("\n\n");
        }

        if (test_for) {
            printf("\nTest for\n\n");

            int lineBreakNum = 1;
            int printedNumCounter = 0;
            for (int i = 1; i < number; i += 2) {
                printf("%d\t", i);
                printedNumCounter++;

                if ((printedNumCounter % lineBreakNum) == 0) {
                    printf("\n");
                    lineBreakNum++;
                    printedNumCounter = 0;
                }
            }

            printf("\n\n");
        }
    }
    else {
        printf("\nnumber must be > 0\n");
    }

    printf("\n************************************************************");

    return 0;
}
```

Test Ausgaben:
(Start values -> „213532“, „a213532“, „213532-“, „213 532“ und „“)

```
#### triangularMatrix ###############################################################

number = 0

number must be > 0

************************************************************
number = 1

Test while



Test doWhile



Test for



************************************************************
number = 2

Test while

1


Test doWhile

1


Test for

1
```

```
************************************************************
number = 20

Test while

1
3       5
7       9       11
13      15      17      19


Test doWhile

1
3       5
7       9       11
13      15      17      19


Test for

1
3       5
7       9       11
13      15      17      19


************************************************************
number = 15

Test while

1
3       5
7       9       11
13


Test doWhile

1
3       5
7       9       11
13


Test for

1
3       5
7       9       11
13
```

```
************************************************************
number = 200

Test while

1
3       5
7       9       11
13      15      17      19
21      23      25      27      29
31      33      35      37      39      41
43      45      47      49      51      53      55
57      59      61      63      65      67      69      71
73      75      77      79      81      83      85      87      89
91      93      95      97      99      101     103     105     107     109
111     113     115     117     119     121     123     125     127     129     131
133     135     137     139     141     143     145     147     149     151     153     155
157     159     161     163     165     167     169     171     173     175     177     179     181
183     185     187     189     191     193     195     197     199

Test doWhile

1
3       5
7       9       11
13      15      17      19
21      23      25      27      29
31      33      35      37      39      41
43      45      47      49      51      53      55
57      59      61      63      65      67      69      71
73      75      77      79      81      83      85      87      89
91      93      95      97      99      101     103     105     107     109
111     113     115     117     119     121     123     125     127     129     131
133     135     137     139     141     143     145     147     149     151     153     155
157     159     161     163     165     167     169     171     173     175     177     179     181
183     185     187     189     191     193     195     197     199

Test for

1
3       5
7       9       11
13      15      17      19
21      23      25      27      29
31      33      35      37      39      41
43      45      47      49      51      53      55
57      59      61      63      65      67      69      71
73      75      77      79      81      83      85      87      89
91      93      95      97      99      101     103     105     107     109
111     113     115     117     119     121     123     125     127     129     131
133     135     137     139     141     143     145     147     149     151     153     155
157     159     161     163     165     167     169     171     173     175     177     179     181
183     185     187     189     191     193     195     197     199

************************************************************
##############################################################################################
```

3 Zinsrechnung

Lösungsidee:
Zuerts werden die Übergabeparameter überprüft. Wenn alle valid sind wird, je nach dem wie viele Jahre man angegeben hat, jedes Jahr zum Startkapital der entsprechende Zinssatz hinzugefügt. Am Ende wird dann das Endkapital ausgegeben.

Testfälle: (1000.00, 5, 10) (-1000.00, 2, 5) (0.00, 10, 3) (666.66, 20, 15) (666.66, 0, 0)

C-Program code:

Int main:

```c
int main(int argc, char* argv[]) {
    int test_ascii2int = 0;
    int test_triangularMatrix = 0;
    int test_interestBill = 1;
    int test_arithmeticAverage = 0;

    if (test_ascii2int) { ... }

    if (test_triangularMatrix) { ... }

    if (test_interestBill) {
        printf("\n\n#### interestBill ########################################################################\n\n");

        // test 1 ---------------
        interestBill(1000.00, 5, 10);
        // test 2 ---------------
        interestBill(-1000.00, 2, 5);
        // test 3 ---------------
        interestBill(0.00, 10, 3);
        // test 4 ---------------
        interestBill(666.66, 20, 15);
        // test 5 ---------------
        interestBill(666.66, 0, 0);
    }

    if (test_arithmeticAverage) { ... }


    // just to make it more beautiful ------------------------------------------------------------------------------------
    printf("\n#############################################################################################\n\n");
    // ------------------------------------------------------------------------------------------------------------------

    return 0;
}
```

Int interestBill:

```c
int interestBill(float capital, int fixedRateInPercent, int years) {
    if (years < 0) {
        years = 0;
    }
    if (fixedRateInPercent < 0) {
        fixedRateInPercent = 0;
    }

    printf("\nCapital development for share capital: %.2f EUR\n", capital);
    printf("Fixed rate: %d%, Duration: %d Years\n\n", fixedRateInPercent, years);

    printf("Year \t\tCapital");
    printf("\n---------------------------------------");
    for (int i = 1; i <= years; i++) {
        capital += capital / 100 * fixedRateInPercent;
        printf("\n%d \t\t%.2f EUR\n", i, capital);
    }

    printf("\nYour capital after %d year(s) is %.2f EUR", years, capital);
    printf("\n*************************************************************");

    return 0;
}
```

Test Ausgaben:
(Start values -> (1000.00, 5, 10) (-1000.00, 2, 5) (0.00, 10, 3) (666.66, 20, 15) (666.66, 0, 0))

```
#### interestBill ################################################################################


Capital development for share capital: 1000.00 EUR
Fixed rate: 5, Duration: 10 Years

Year            Capital
----------------------------------------
1               1050.00 EUR

2               1102.50 EUR

3               1157.63 EUR

4               1215.51 EUR

5               1276.28 EUR

6               1340.10 EUR

7               1407.10 EUR

8               1477.46 EUR

9               1551.33 EUR

10              1628.89 EUR

Your capital after 10 year(s) is 1628.89 EUR
****************************************************************
Capital development for share capital: -1000.00 EUR
Fixed rate: 2, Duration: 5 Years

Year            Capital
----------------------------------------
1               -1020.00 EUR

2               -1040.40 EUR

3               -1061.21 EUR

4               -1082.43 EUR

5               -1104.08 EUR

Your capital after 5 year(s) is -1104.08 EUR
```

```
****************************************************************
Capital development for share capital: 0.00 EUR
Fixed rate: 10, Duration: 3 Years

Year            Capital
----------------------------------------
1               0.00 EUR

2               0.00 EUR

3               0.00 EUR

Your capital after 3 year(s) is 0.00 EUR
****************************************************************
Capital development for share capital: 666.66 EUR
Fixed rate: 20, Duration: 15 Years

Year            Capital
----------------------------------------
1               799.99 EUR

2               959.99 EUR

3               1151.99 EUR

4               1382.39 EUR

5               1658.86 EUR

6               1990.64 EUR

7               2388.76 EUR

8               2866.52 EUR

9               3439.82 EUR

10              4127.78 EUR

11              4953.34 EUR

12              5944.01 EUR

13              7132.81 EUR

14              8559.37 EUR

15              10271.24 EUR

Your capital after 15 year(s) is 10271.24 EUR
```

```
****************************************************************
Capital development for share capital: 666.66 EUR
Fixed rate: 0, Duration: 0 Years

Year            Capital
----------------------------------------
Your capital after 0 year(s) is 666.66 EUR
****************************************************************
################################################################################################
```

4 Arithmetisches Mittel + Ausreißer

Lösungsidee:
Zuerts werden die Übergabeparameter überprüft. Die länge der Arrays wird mit ‚#define' definiert. Die min/max Variabeln werden mit dem ersten Wert vom array intialisiert damit die min/max Variabeln einen richtigen Anfangswert haben. Dannach wird mit der vordefinierten länge vom Array durch das Array iteriert und in jeder Iteration wird überprüft ob es ein neuen min/max Wert gibt. Am Ende der Schleife wird der aktuelle Wert in einer von außerhalb der Schleife deklarierte Variable hinzugezählt.

Testfälle: { 3, 5, 7, 8, 1, -1, 4, 0 } { 0, 4, 1, -10, 4, 0 } { 2 } { }

C-Program code:

#defines:

```
#define NR_OF_VALUES_8 8
#define NR_OF_VALUES_6 6
#define NR_OF_VALUES_1 1
#define NR_OF_VALUES_0 0
```

Int main

```
int main(int argc, char* argv[]) {
    int test_ascii2int = 0;
    int test_triangularMatrix = 0;
    int test_interestBill = 0;
    int test_arithmeticAverage = 1;

    if (test_ascii2int) { ... }

    if (test_triangularMatrix) { ... }

    if (test_interestBill) { ... }

    if (test_arithmeticAverage) {
        printf("\n\n#### arithmeticAverage ##########################################################################\n\n");

        // test 1 ----------------
        {
            int values[] = { 3, 5, 7, 8, 1, -1, 4, 0 };
            printf("\nvalues: { 3, 5, 7, 8, 1, -1, 4, 0 }");
            arithmeticAverage(NR_OF_VALUES_8, values);
        }
        // test 2 ----------------
        {
            int values[] = { 0, 4, 1, -10, 4, 0 };
            printf("\nvalues: { 0, 4, 1, -10, 4, 0 }");
            arithmeticAverage(NR_OF_VALUES_6, values);
        }
        // test 3 ----------------
        {
            int values[] = { 2 };
            printf("\nvalues: { 2 }");
            arithmeticAverage(NR_OF_VALUES_1, values);
        }
        // test 4 ----------------
        {
            printf("\nvalues: { }");
            arithmeticAverage(NR_OF_VALUES_0, NULL);
        }

    }


    // just to make it more beautiful ------------------------------------------------------------------------------------------
    printf("\n###############################################################################################\n\n");
    // -------------------------------------------------------------------------------------------------------------------------

    return 0;
}
```

Int arithmeticAverage:

```c
int arithmeticAverage(int valueCount, int values[]) {
    printf("\nvalueCount: %d\n", valueCount);

    if (valueCount > 0) {
        int min = values[0];
        int max = values[0];
        int arithAvg = 0;

        printf("\ncalculating min, max and arithmeticAverage...");
        for (int i = 0; i < valueCount; i++) {
            int currVal = values[i];

            if (currVal < min) {
                min = currVal;
            }
            if (currVal > max) {
                max = currVal;
            }

            arithAvg += currVal;
        }
        printf("\nfinished\n");

        printf("\nArithmetic average: %.3f", (float)arithAvg / valueCount);
        printf("\nmin: %d", min);
        printf("\nmax: %d", max);
    }
    else {
        printf("\narray is empty!");
    }

    printf("\n************************************************************");

    return 0;
}
```

Test Ausgaben:
(Start values -> { 3, 5, 7, 8, 1, -1, 4, 0 } { 0, 4, 1, -10, 4, 0 } { 2 } { })

```
#### arithmeticAverage ###########################################################


values: { 3, 5, 7, 8, 1, -1, 4, 0 }
valueCount: 8

calculating min, max and arithmeticAverage...
finished

Arithmetic average: 3.375
min: -1
max: 8
************************************************************
values: { 0, 4, 1, -10, 4, 0 }
valueCount: 6

calculating min, max and arithmeticAverage...
finished

Arithmetic average: -0.167
min: -10
max: 4
************************************************************
values: { 2 }
valueCount: 1

calculating min, max and arithmeticAverage...
finished

Arithmetic average: 2.000
min: 2
max: 2
************************************************************
values: { }
valueCount: 0

array is empty!
************************************************************
##############################################################################################
```