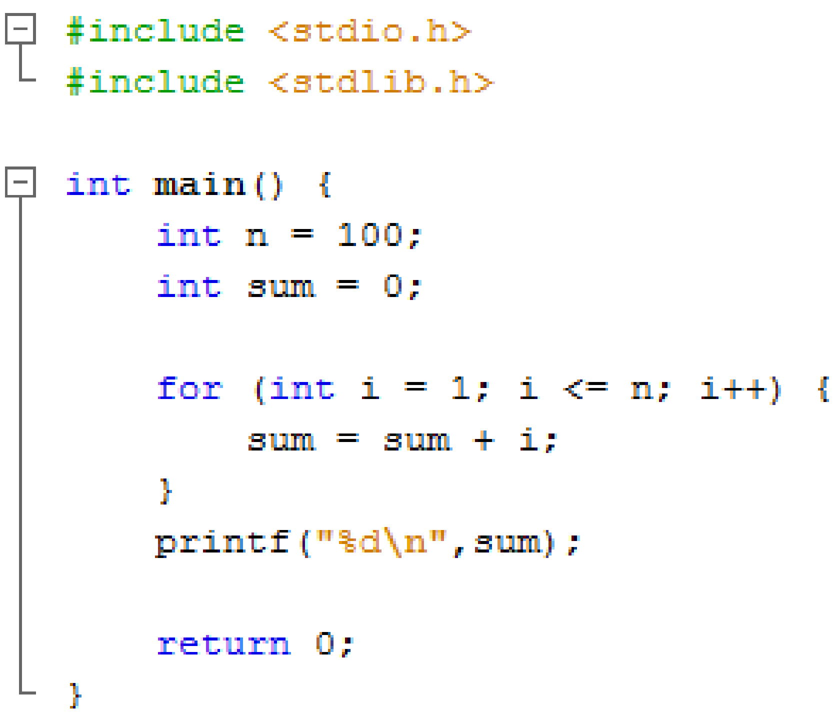
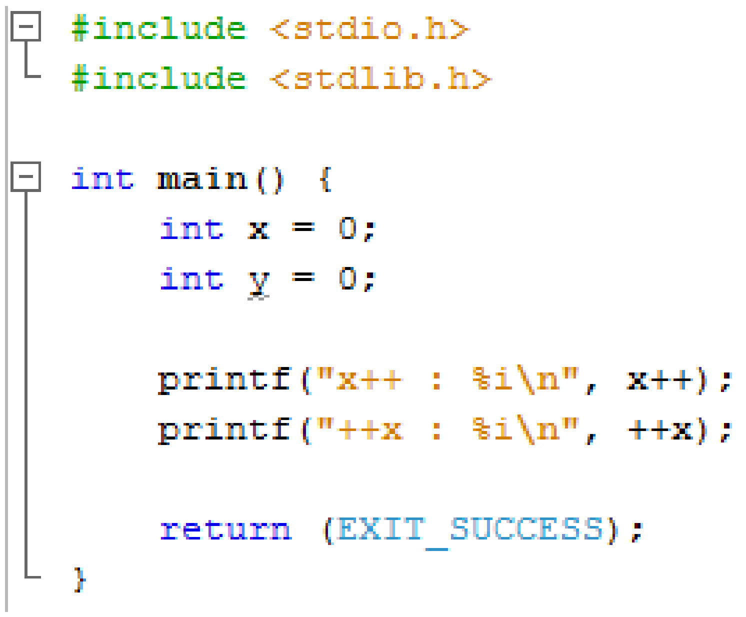
Programme vom Praktikum 1

Arithmetische Summe



```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int n = 100;
    int sum = 0;

    for (int i = 1; i <= n; i++) {
        sum = sum + i;
    }
    printf("%d\n",sum);

    return 0;
}
```

Post-Inkrement

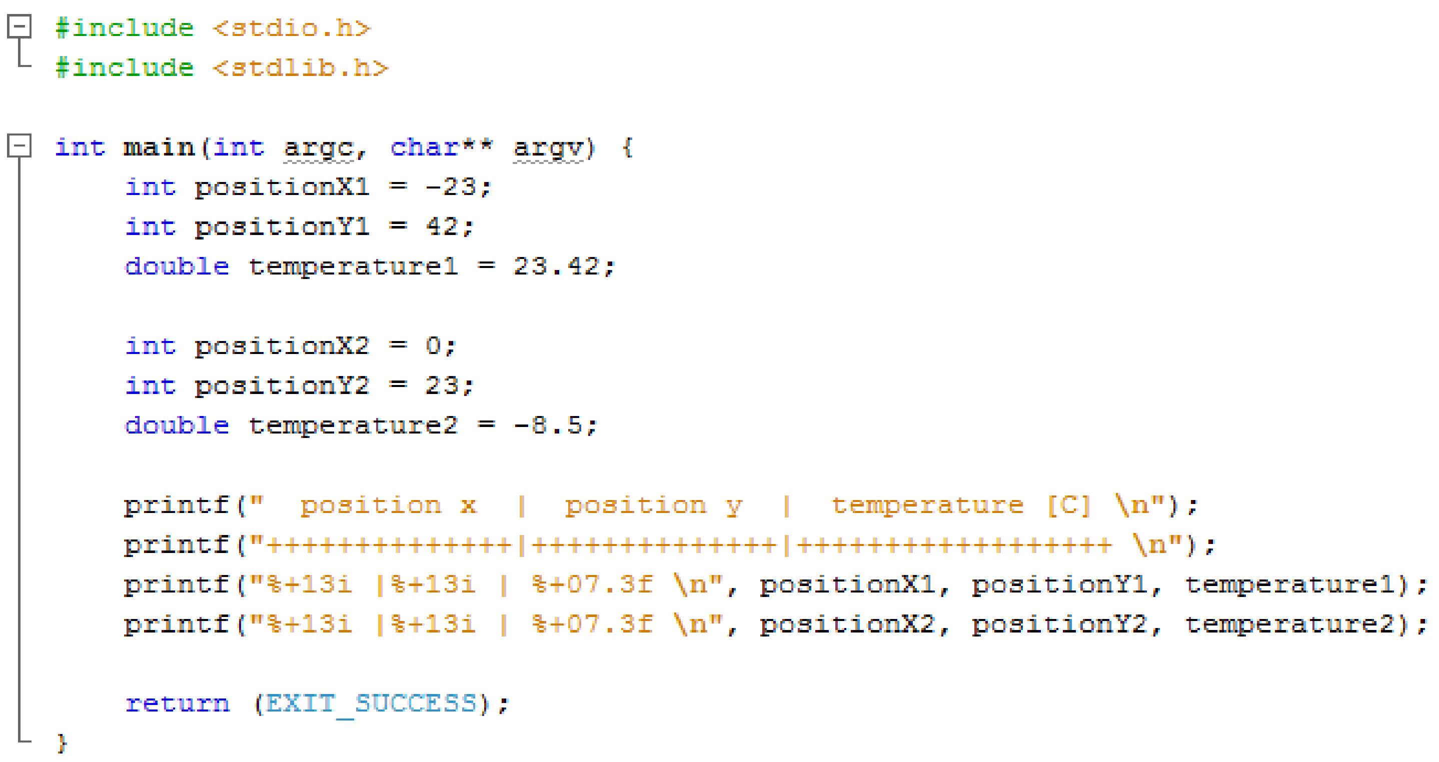


```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int x = 0;
    int y = 0;

    printf("x++ : %i\n", x++);
    printf("++x : %i\n", ++x);

    return (EXIT_SUCCESS);
}
```

Ausgabe:

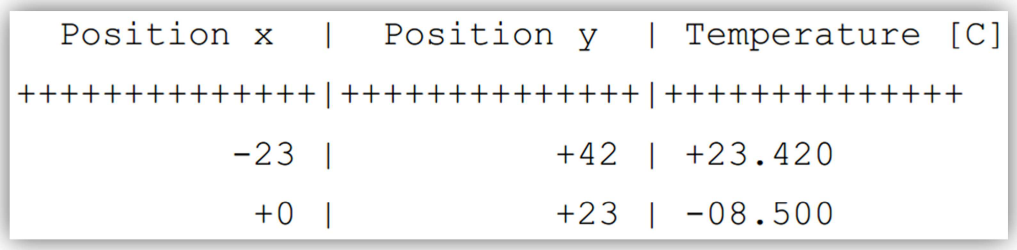
```
x++ : 0
++x : 2
```

Tabellenausgabe



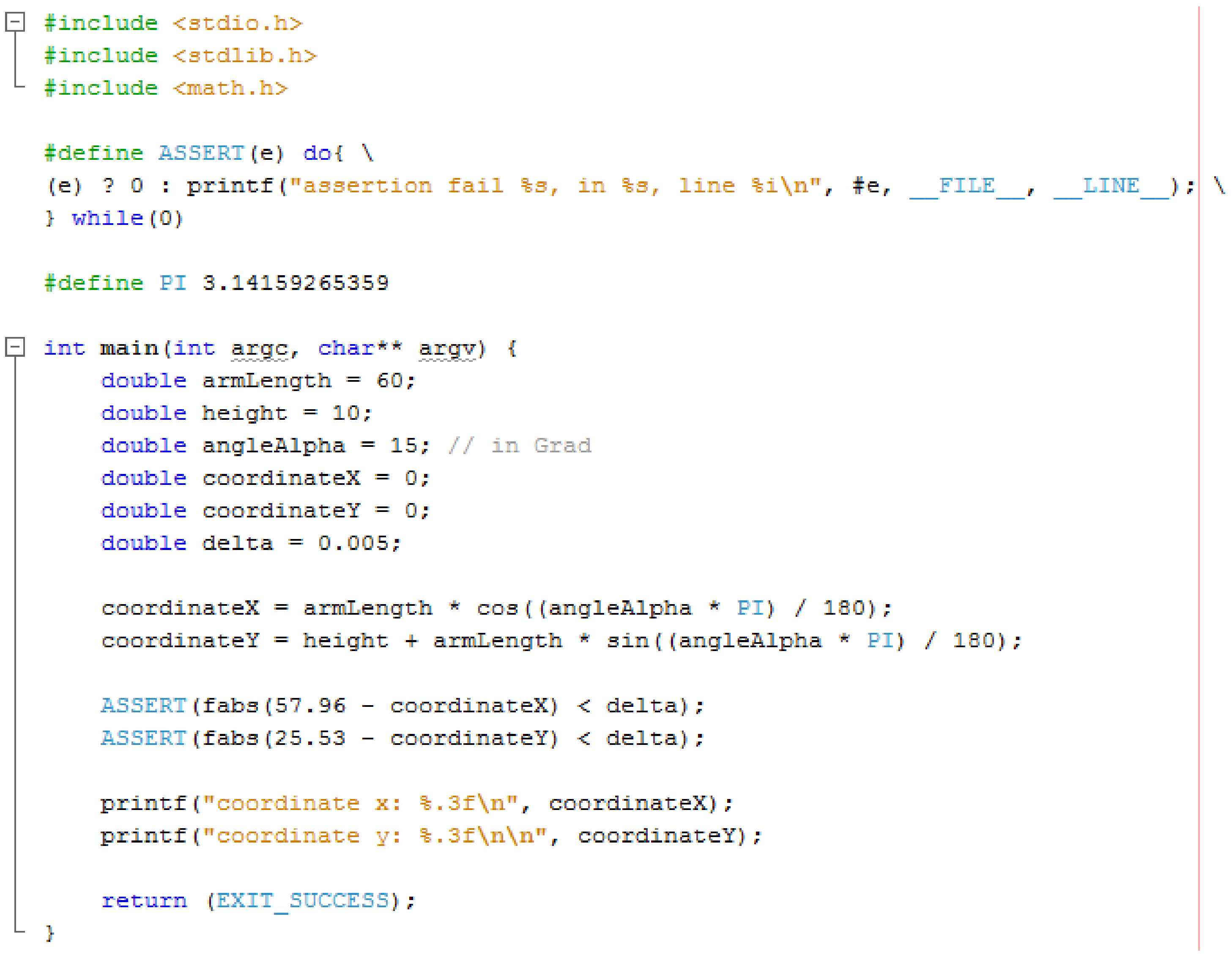
```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
    int positionX1 = -23;
    int positionY1 = 42;
    double temperature1 = 23.42;

    int positionX2 = 0;
    int positionY2 = 23;
    double temperature2 = -8.5;

    printf("  position x   |  position y   |  temperature [C] \n");
    printf("++++++++++++++|++++++++++++++|++++++++++++++++++ \n");
    printf("%+13i |%+13i | %+07.3f \n", positionX1, positionY1, temperature1);
    printf("%+13i |%+13i | %+07.3f \n", positionX2, positionY2, temperature2);

    return (EXIT_SUCCESS);
}
```

Ausgabe:



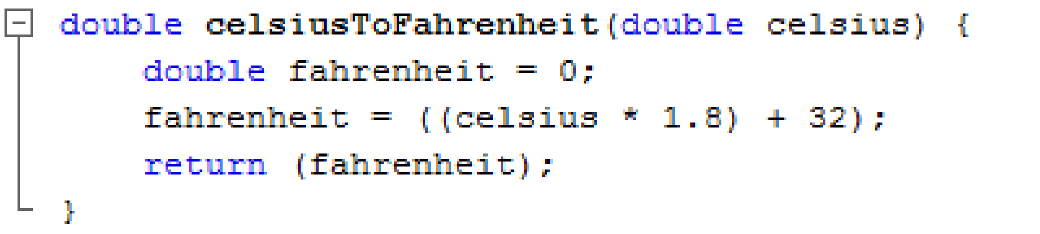
```
  Position x  |  Position y  | Temperature [C]
++++++++++++++|++++++++++++++|++++++++++++++
          -23 |          +42 | +23.420
           +0 |          +23 | -08.500
```

Positionsberechnung vom Roboterarm



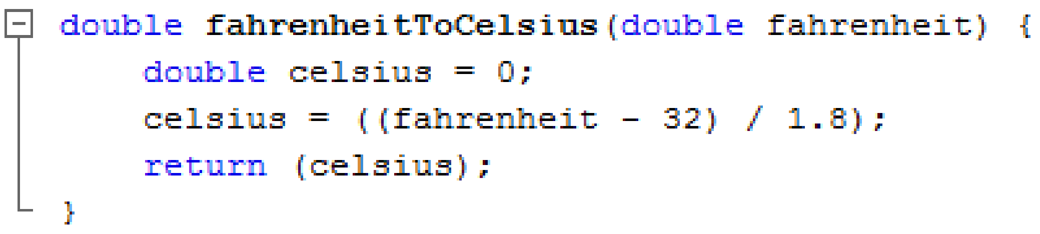
```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define ASSERT(e) do{ \
(e) ? 0 : printf("assertion fail %s, in %s, line %i\n", #e, __FILE__, __LINE__); \
} while(0)

#define PI 3.14159265359

int main(int argc, char** argv) {
    double armLength = 60;
    double height = 10;
    double angleAlpha = 15; // in Grad
    double coordinateX = 0;
    double coordinateY = 0;
    double delta = 0.005;

    coordinateX = armLength * cos((angleAlpha * PI) / 180);
    coordinateY = height + armLength * sin((angleAlpha * PI) / 180);

    ASSERT(fabs(57.96 - coordinateX) < delta);
    ASSERT(fabs(25.53 - coordinateY) < delta);

    printf("coordinate x: %.3f\n", coordinateX);
    printf("coordinate y: %.3f\n\n", coordinateY);

    return (EXIT_SUCCESS);
}
```

Funktionen zum Temperaturen umrechnen



```
double celsiusToFahrenheit(double celsius) {
    double fahrenheit = 0;
    fahrenheit = ((celsius * 1.8) + 32);
    return (fahrenheit);
}
```

°C -> °F



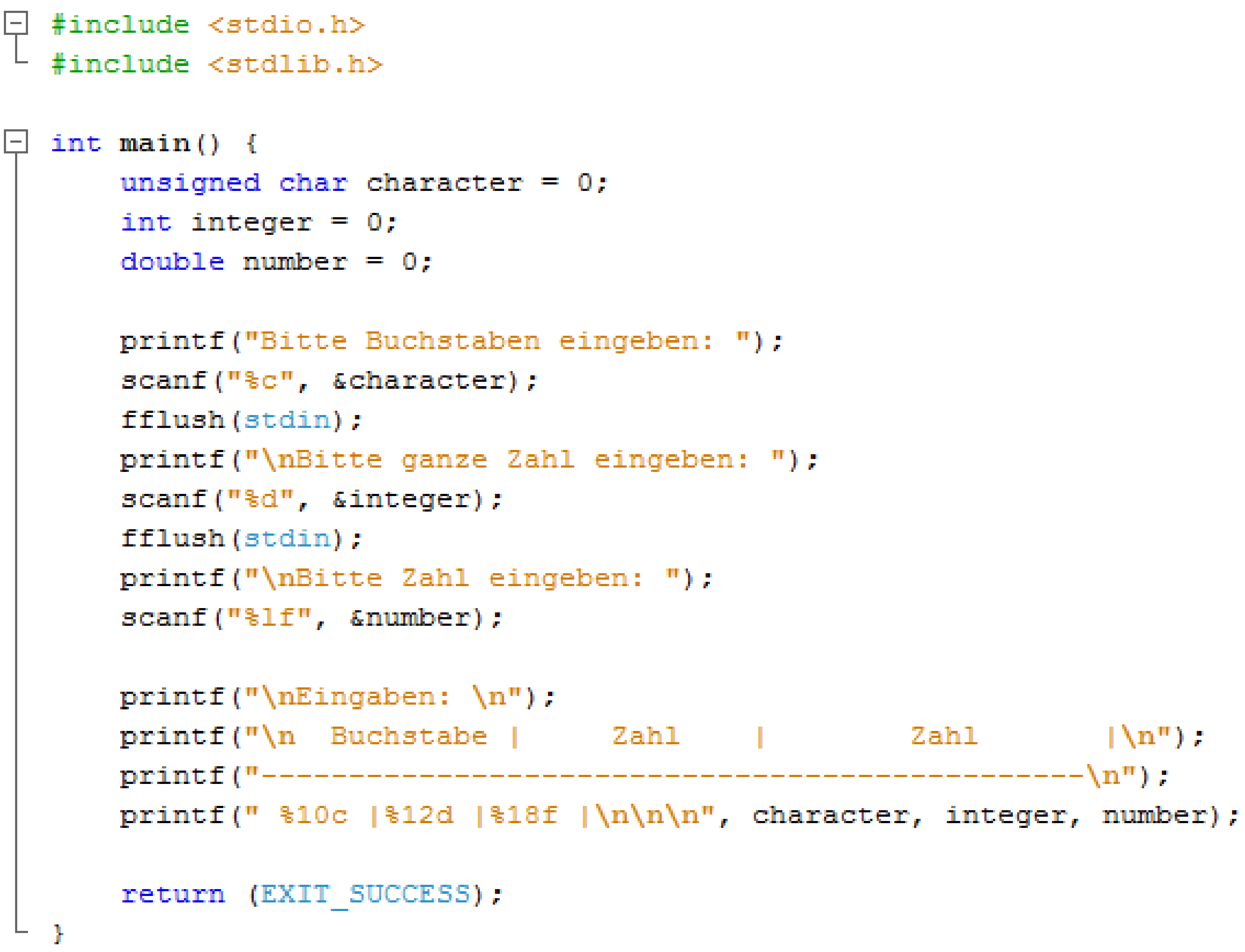
```
double fahrenheitToCelsius(double fahrenheit) {
    double celsius = 0;
    celsius = ((fahrenheit - 32) / 1.8);
    return (celsius);
}
```

°F -> °C

$$F = (C * 1{,}8) + 32$$

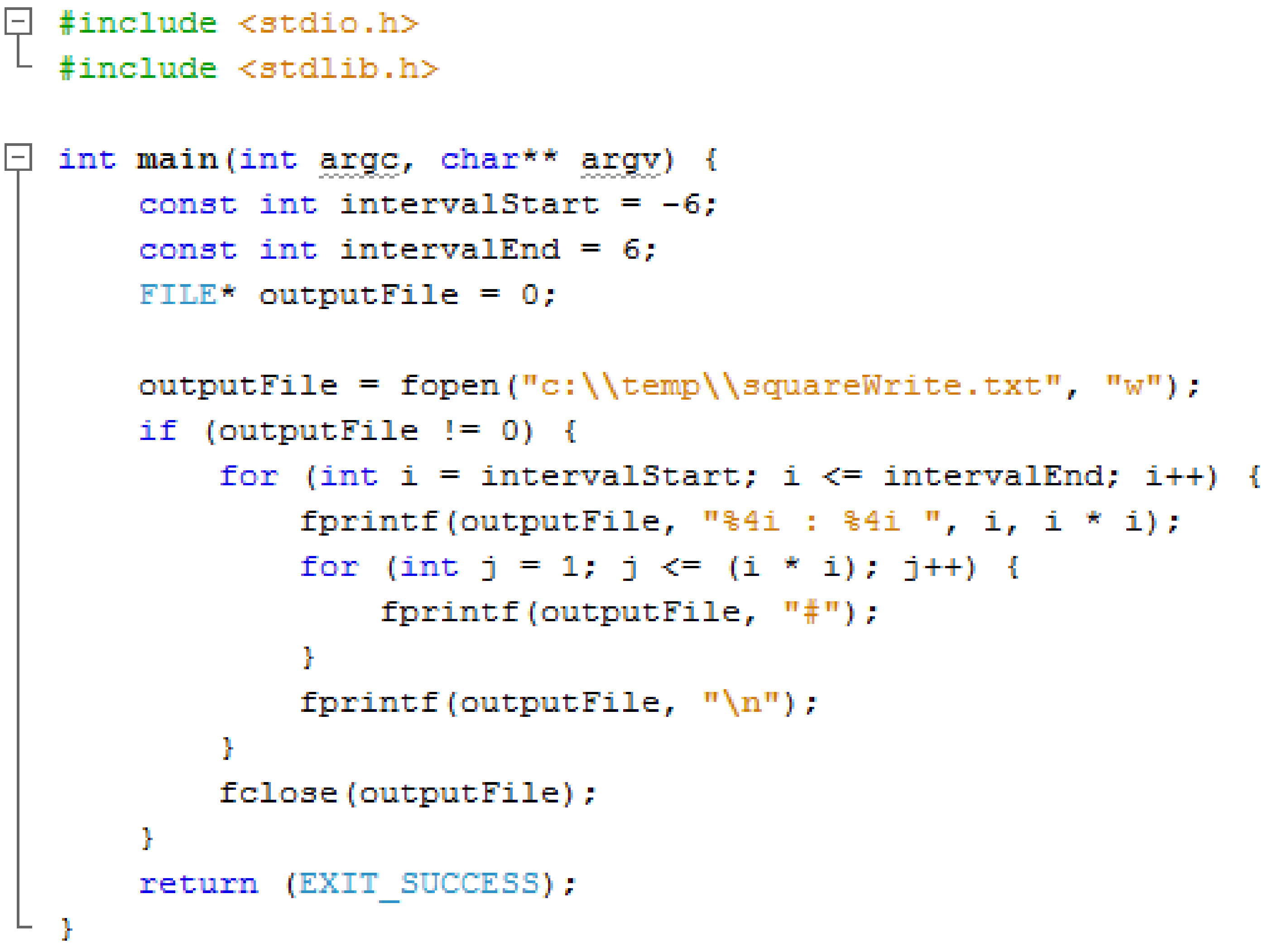
$$C = \frac{F - 32}{1{,}8}$$

scanf mit Tabellenausgabe



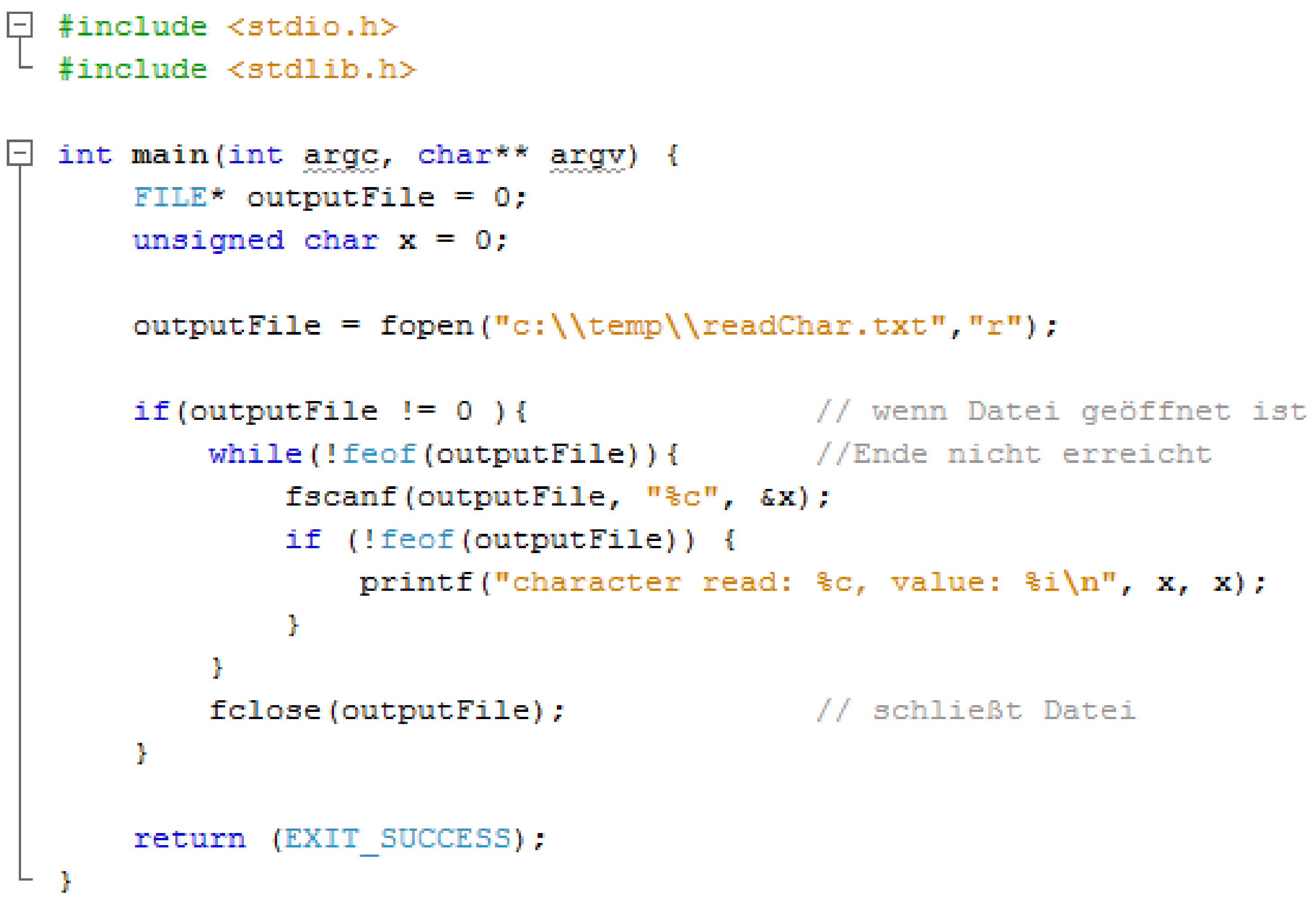
```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    unsigned char character = 0;
    int integer = 0;
    double number = 0;

    printf("Bitte Buchstaben eingeben: ");
    scanf("%c", &character);
    fflush(stdin);
    printf("\nBitte ganze Zahl eingeben: ");
    scanf("%d", &integer);
    fflush(stdin);
    printf("\nBitte Zahl eingeben: ");
    scanf("%lf", &number);

    printf("\nEingaben: \n");
    printf("\n  Buchstabe |     Zahl    |        Zahl       |\n");
    printf("-----------------------------------------------\n");
    printf(" %10c |%12d |%18f |\n\n\n", character, integer, number);

    return (EXIT_SUCCESS);
}
```

TXT-Datei beschreiben



```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
    const int intervalStart = -6;
    const int intervalEnd = 6;
    FILE* outputFile = 0;

    outputFile = fopen("c:\\temp\\squareWrite.txt", "w");
    if (outputFile != 0) {
        for (int i = intervalStart; i <= intervalEnd; i++) {
            fprintf(outputFile, "%4i : %4i ", i, i * i);
            for (int j = 1; j <= (i * i); j++) {
                fprintf(outputFile, "#");
            }
            fprintf(outputFile, "\n");
        }
        fclose(outputFile);
    }
    return (EXIT_SUCCESS);
}
```

TXT-Datei einlesen

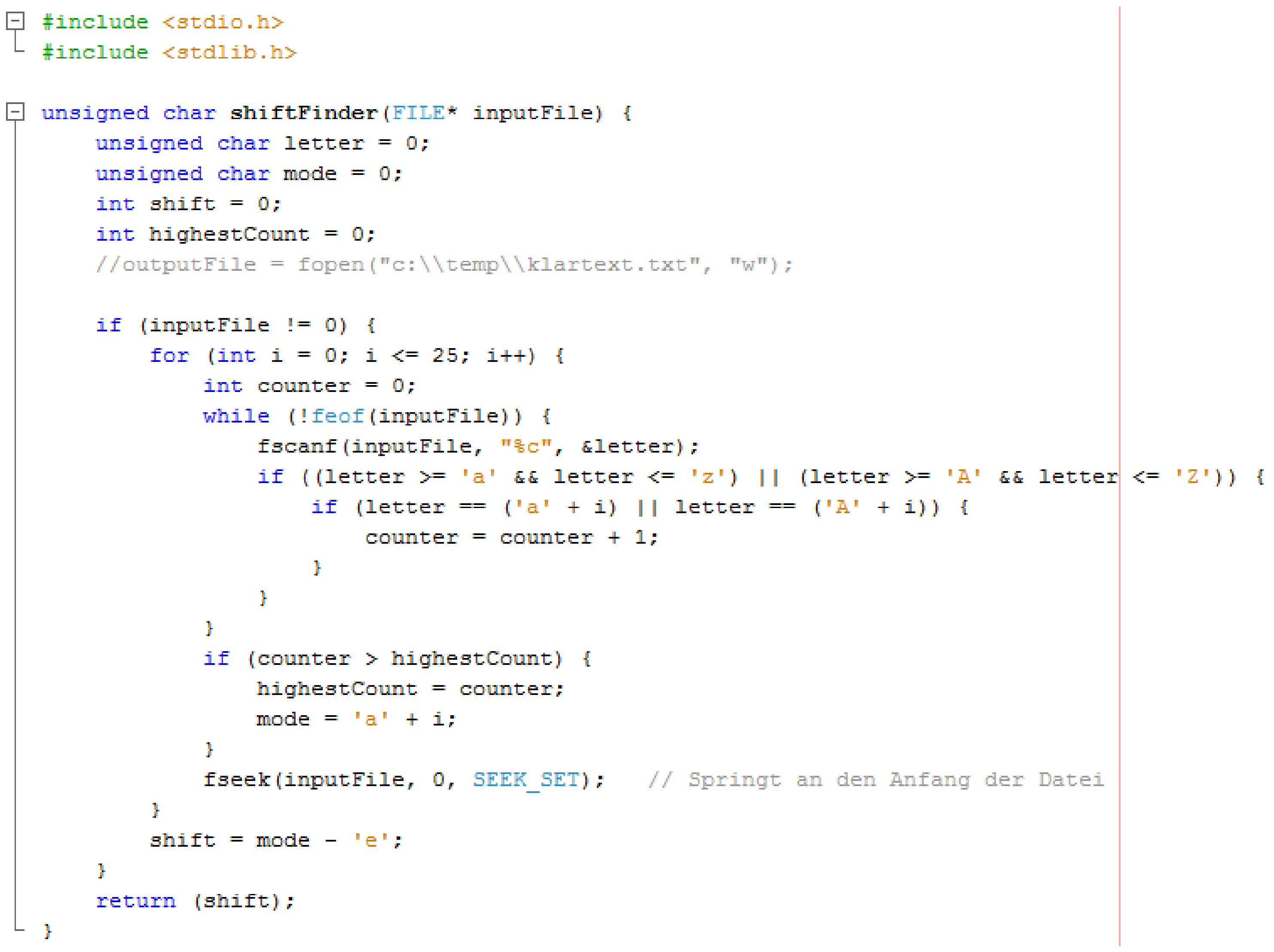


```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
    FILE* outputFile = 0;
    unsigned char x = 0;

    outputFile = fopen("c:\\temp\\readChar.txt","r");

    if(outputFile != 0 ){                   // wenn Datei geöffnet ist
        while(!feof(outputFile)){           //Ende nicht erreicht
            fscanf(outputFile, "%c", &x);
            if (!feof(outputFile)) {
                printf("character read: %c, value: %i\n", x, x);
            }
        }
        fclose(outputFile);                 // schließt Datei
    }

    return (EXIT_SUCCESS);
}
```

Text entschlüsseln

```
#include <stdio.h>
#include <stdlib.h>
#include "crypting.h"

int main(int argc, char** argv) {
    FILE* inputFile = 0;
    FILE* outputFile = 0;
    unsigned char letter = 0;
    int shift = 13;

    inputFile = fopen("c:\\temp\\chiper.txt", "r");
    outputFile = fopen("c:\\temp\\klartext.txt", "w");

    if (inputFile != 0 && outputFile != 0) {
        while (!feof(inputFile)) {
            fscanf(inputFile, "%c", &letter);
            if (!feof(inputFile)) {
                printf("%c", decrypt(letter, shift));
                fprintf(outputFile, "%c", decrypt(letter, shift));
            }
        }
        fclose(inputFile);
        fclose(outputFile);
    }

    return (EXIT_SUCCESS);
}

unsigned char encrypt(unsigned char letter, int shift) {
    if ((letter >= 'a' && letter <= 'z') || (letter >= 'A' && letter <= 'Z')) {
        if (letter > ('Z' - shift) && letter <= 'Z') {
            letter = letter - 26;
        }
        if (letter > ('z' - shift) && letter <= 'z') {
            letter = letter - 26;
        }
        letter = letter + shift;
    }
    return (letter);
}

unsigned char decrypt(unsigned char letter, int shift) {
    if ((letter >= 'a' && letter <= 'z') || (letter >= 'A' && letter <= 'Z')) {
        if (letter >= 'a' && letter < ('a' + shift)) {
            letter = letter + 26;
        }
        if (letter >= 'A' && letter < ('A' + shift)) {
            letter = letter + 26;
        }
        letter = letter - shift;
    }
    return (letter);
}
```

Automatisch Verschiebeweite ermitteln



```c
#include <stdio.h>
#include <stdlib.h>

unsigned char shiftFinder(FILE* inputFile) {
    unsigned char letter = 0;
    unsigned char mode = 0;
    int shift = 0;
    int highestCount = 0;
    //outputFile = fopen("c:\\temp\\klartext.txt", "w");

    if (inputFile != 0) {
        for (int i = 0; i <= 25; i++) {
            int counter = 0;
            while (!feof(inputFile)) {
                fscanf(inputFile, "%c", &letter);
                if ((letter >= 'a' && letter <= 'z') || (letter >= 'A' && letter <= 'Z')) {
                    if (letter == ('a' + i) || letter == ('A' + i)) {
                        counter = counter + 1;
                    }
                }
            }
            if (counter > highestCount) {
                highestCount = counter;
                mode = 'a' + i;
            }
            fseek(inputFile, 0, SEEK_SET);    // Springt an den Anfang der Datei
        }
        shift = mode - 'e';
    }
    return (shift);
}
```

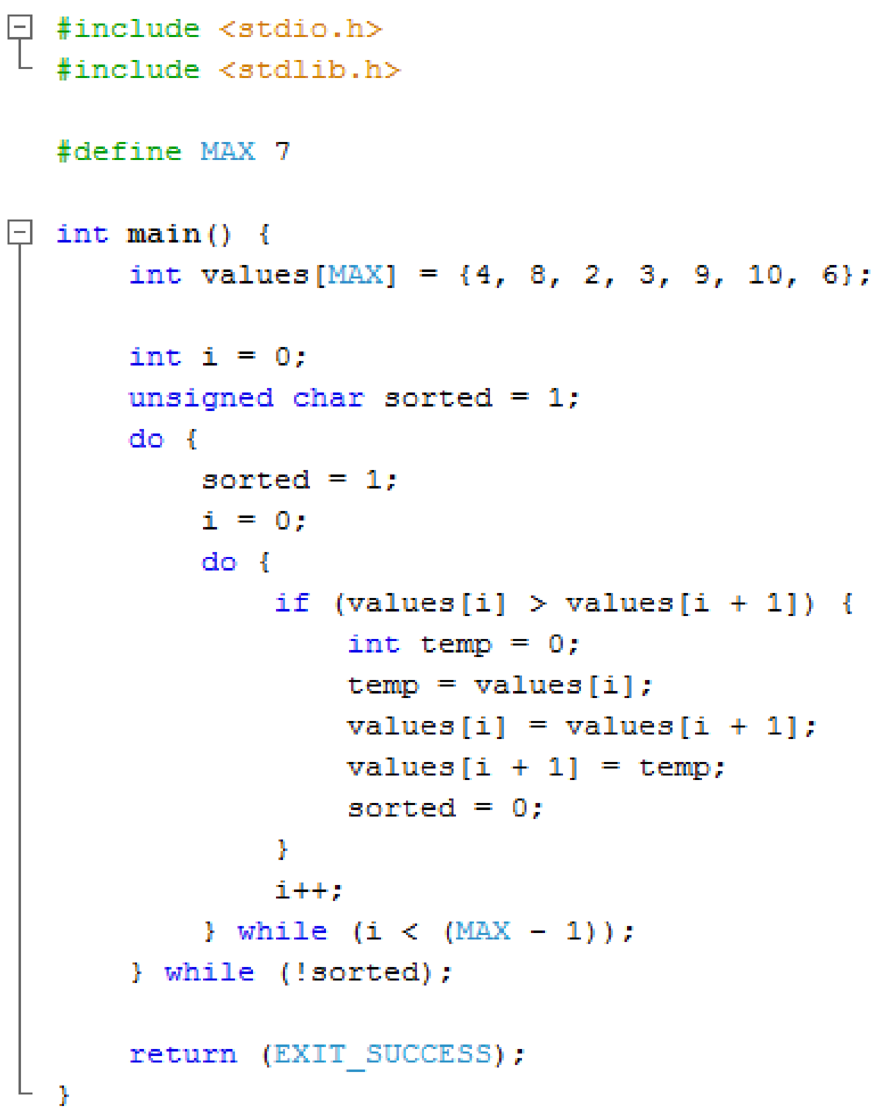
String-Verarbeitung

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char string1[] = "Beispieltext";
    char string2[] = "Folgetext mit Leerzeichen!";
    char string3[80] = {0};

    printf("String1[13] = %s\n\n", string1);

    printf("String1: %s\n", string1);
    printf("String2: %s\n\n", string2);

    printf("Bytes von String-String 1: %d\n", sizeof (string1) / sizeof (string1[0]));      //13
    printf("Bytes von String-String 2: %d\n\n", sizeof (string2) / sizeof (string2[0]));   //17

    printf("Speicherbereich: %X\n", &string1);
    printf("Speicherbereich: %X\n\n", &string2);

    printf("Stringlaenge 1: %d\n", strlen(string1));        //12
    printf("Stringlaenge 2: %d\n\n", strlen(string2));      //26

    string2[26] = 64;
    printf("Null-Terminierung wurde mit 64 ersetzt:\n");
    printf("String2: %s\n", string2);
    printf("Bytes von String 2: %d\n", sizeof (string2) / sizeof (string2[0]));  //27
    printf("Stringlaenge 2: %d\n\n", strlen(string2));                            //39
    string2[26] = 0;

    printf("String1: %s\n", string1);
    printf("String2: %s\n\n", string2);

    printf("strcmp(1,1): %d\n",strcmp(string1,string1));        //  0
    printf("strcmp(2,2): %d\n",strcmp(string2,string2));        //  0
    printf("strcmp(1,2): %d\n",strcmp(string1,string2));        // -1
    printf("strcmp(2,1): %d\n\n",strcmp(string2,string1));      //  1

    printf("String1: %s\n", string1);
    printf("String2: %s\n\n", string2);

    printf("strcpy(1,2):\n");
    strcpy(string1,string2);        // kopiert string2 nach string1
    printf("String1: %s\n", string1);
    printf("String2: %s\n\n", string2);

    printf("Strings verketten(string1 -> string3):\n");
    strcpy(string3,string1);
    printf("strcat(3,2): %s\n\n",strcat(string3,string2));

    return (EXIT_SUCCESS);
}
```

Größte Element eines Arrays

```c
#include <stdio.h>

unsigned int posMaxOfArray(int list[], unsigned int sizeOfArray);

int main() {
    int list1[] = {101, 2, 34, 3, 1, 31, 9, 100, 200};
    unsigned int sizeOfArray = 0;

    sizeOfArray = sizeof (list1) / sizeof (list1[0]);

    printf("groesste Zahl: %d\n", list1[posMaxOfArray(list1, sizeOfArray)]);
    printf("Position: %d\n", posMaxOfArray(list1, sizeOfArray));

    return 0;
}

unsigned int posMaxOfArray(int list[], unsigned int sizeOfArray) {
    unsigned char position = 0;
    int maxValue = 0;
    for (int i = 0; i < sizeOfArray; i++) {
        if (list[i] > maxValue) {
            maxValue = list[i];
            position = i;
        }
    }
    return (position);
}
```

Sortier-Algorythmus

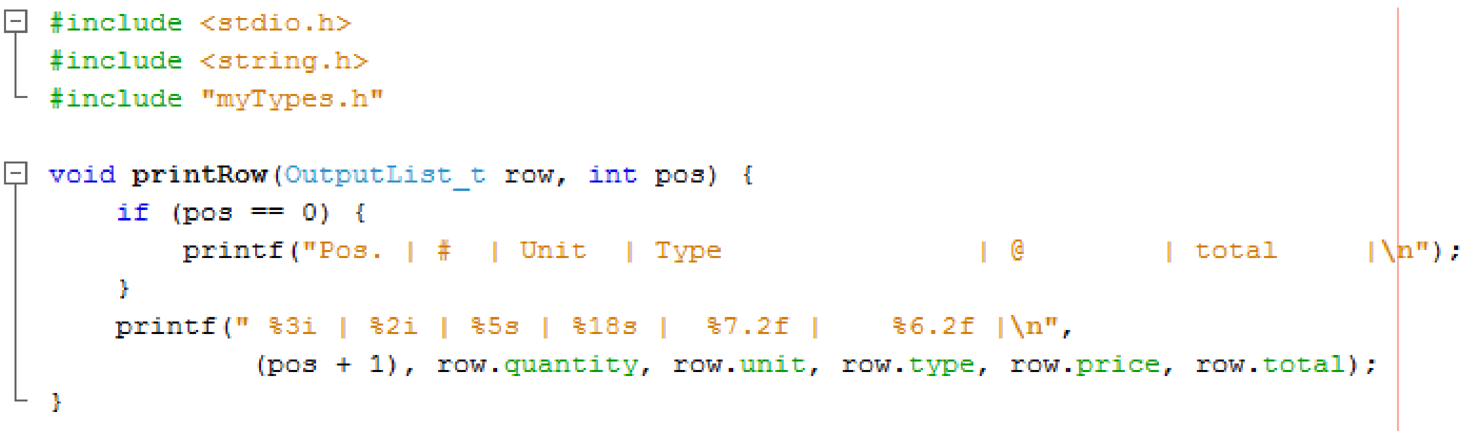


```c
#include <stdio.h>
#include <stdlib.h>

#define MAX 7

int main() {
    int values[MAX] = {4, 8, 2, 3, 9, 10, 6};

    int i = 0;
    unsigned char sorted = 1;
    do {
        sorted = 1;
        i = 0;
        do {
            if (values[i] > values[i + 1]) {
                int temp = 0;
                temp = values[i];
                values[i] = values[i + 1];
                values[i + 1] = temp;
                sorted = 0;
            }
            i++;
        } while (i < (MAX - 1));
    } while (!sorted);

    return (EXIT_SUCCESS);
}
```

Stückliste mit Arrays und typedef

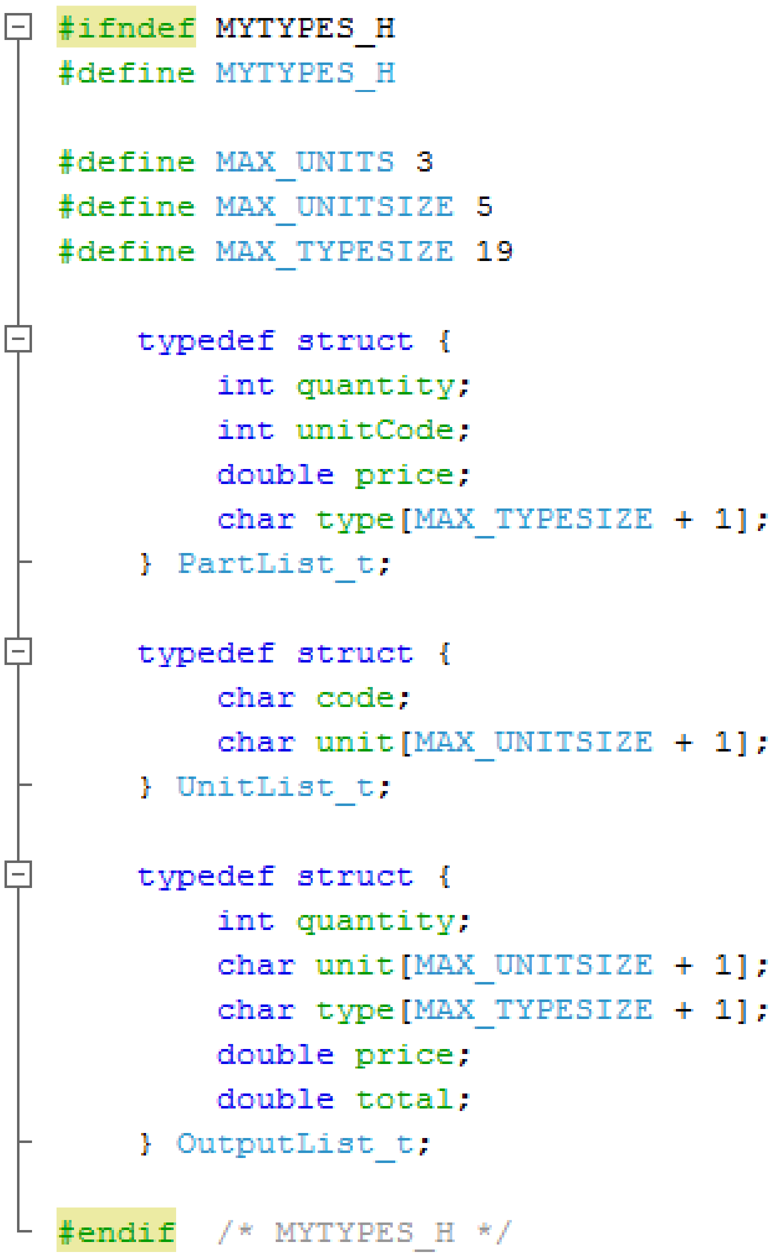
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "printRow.h"
#include "myTypes.h"

int main(int argc, char** argv) {
    PartList_t partList = {0};

    UnitList_t unitList[MAX_UNITS] = {
        {23, "m"},
        {24, "pice"},
        {55, "ml"},
    };

    OutputList_t outputList;
    FILE* partListFile = 0;

    partListFile = fopen("c:\\temp\\data.csv", "r");

    if (partListFile != 0) { // wenn Datei geöffnet ist
        int pos = 0;

        while (!feof(partListFile)) { //solange Ende nicht erreicht
            fscanf(partListFile, "%i;%d;%lf;%s\n", &partList.quantity,
                    &partList.unitCode, &partList.price, (partList.type));

            outputList.quantity = partList.quantity;

            for (int k = 0; k < MAX_UNITS; k++) {
                if (partList.unitCode == unitList[k].code) {
                    strncpy(outputList.unit, unitList[k].unit, MAX_UNITSIZE);
                }
            }
            strncpy(outputList.type, partList.type, MAX_TYPESIZE);
            outputList.price = partList.price;
            outputList.total = partList.quantity * partList.price;

            printRow(outputList, pos);

            pos++;
        }
        fclose(partListFile); // schließt Datei
    }

    printf("\n");
    return (EXIT_SUCCESS);
}
```

printRow.c



```c
#include <stdio.h>
#include <string.h>
#include "myTypes.h"

void printRow(OutputList_t row, int pos) {
    if (pos == 0) {
        printf("Pos. | #  | Unit  | Type               | @         | total     |\n");
    }
    printf(" %3i | %2i | %5s | %18s |  %7.2f |    %6.2f |\n",
            (pos + 1), row.quantity, row.unit, row.type, row.price, row.total);
}
```

myTypes.h



```c
#ifndef MYTYPES_H
#define MYTYPES_H

#define MAX_UNITS 3
#define MAX_UNITSIZE 5
#define MAX_TYPESIZE 19

    typedef struct {
        int quantity;
        int unitCode;
        double price;
        char type[MAX_TYPESIZE + 1];
    } PartList_t;

    typedef struct {
        char code;
        char unit[MAX_UNITSIZE + 1];
    } UnitList_t;

    typedef struct {
        int quantity;
        char unit[MAX_UNITSIZE + 1];
        char type[MAX_TYPESIZE + 1];
        double price;
        double total;
    } OutputList_t;

#endif  /* MYTYPES_H */
```

Pointer

```
// Deklaration
int *ptr;
int var, var2;

// Initialisieren: ptr bekommt die Adresse von var.
ptr =& var;

// Dereferenzierung : var bekommt den Wert 100 zugewiesen.
*ptr=100;


// var2 mit demselben Wert wie var initialisieren
var2 = *ptr;

*ptr+=100;      // Dereferenzierung: var wird um 100 erhöht.
(*ptr)++;       // Dereferenzierung: var hat jetzt den Wert 201.
(*ptr)--;       // var hat wieder den Wert 200.
ptr=&var2;      // ptr zeigt auf var2.

printf("%d", *ptr);    // Gibt Wert von var2 aus.
printf("%p", &ptr);    // Gibt Adresse von ptr aus.
printf("%p", ptr);     // Gibt Adresse von var2 aus.
```

Pointer und Arrays

Zugriff

```
int array[10];             // Deklaration
int *pointer1, *pointer2;
pointer1 = array;          // pointer1 auf Anfangsadresse von array
pointer2 = array + 3;      // pointer2 auf 4.Element von array

array[0]       = 99;     // array[0]
pointer1[1]    = 88;     // array[1]
*(pointer1+2)  = 77;     // array[2]
*pointer2      = 66;     // array[3]
```

Funktionsparameter

```
int werte[] = { 1, 2, 3, 5, 8 };
int *pointer;
pointer = werte;

funktion(werte);         // 1. Möglichkeit
funktion(&werte[0]);     // 2. Möglichkeit
funktion(pointer);       // 3. Möglichkeit
funktion(&werte[2]);     // Adresse vom 3.Element an funktion
```

Bsp: Pointer als Funktionsparameter

```
#include <stdio.h>
#include <stdlib.h>

void funktion(int *array, int n_array) {
   int i;

   for(i=0; i < n_array; i++)
      printf("%d ",array[i]);
   printf("\n");
}

int main(void) {
   int werte[] = { 1, 2, 3, 5, 8, 13, 21 };

   funktion(werte, sizeof(werte) / sizeof(int));
   return EXIT_SUCCESS;
}
```

Array als Funktions-Rückgabewert

```c
#include <stdio.h>
#include <stdlib.h>

struct array{ int wert[3]; };

struct array init_array(void) {
   int i;
   struct array z;

   for(i = 0; i < sizeof(struct array) / sizeof(int); i++) {
      printf("Wert %d eingeben: ",i);
      scanf("%d",&z.wert[i]);
   }
   return z;
}

void output_array(struct array z) {
   int i;

   for(i = 0; i < sizeof(struct array) / sizeof(int); i++)
      printf("%d\t", z.wert[i]);
   printf("\n");
}

int main(void) {
   struct array new_array;

   /* Array als Rückgabewert in einer Struktur verschachtelt */
   new_array=init_array();
   /* call-by-value */
   output_array(new_array);
   return EXIT_SUCCESS;
}
```