OEFENINGEN ARRAYS

1. Schrijf een programma die een array van 5 elementen allemaal op 0 zet.

2. Schrijf een programma die een array met een lijst (met getallen) initialiseert.

3. Schrijf een programma die de waarden van een array bij elkaar optelt.

4. 40 Studenten zijn gevraagd om een enquête in te vullen om de kwaliteit van de worstenbroodjes in de Jumbo te meten. Ze konden de kwaliteit aangeven op een schaal van 1 tot 10 (1 = afschuwelijk, 10 = fantastisch). Zet de 40 uitkomsten in een integer array en geef aan hoe vaak de kwaliteit is gemeten (dus hoe vaak is 1 gekozen, hoe vaak 2, hoe vaak 3, enz.).

5. Schrijf een programma die een dobbelsteen 120.000.000 keer gooit. Hou bij hoe vaak telkens een zijde is gegooid. Hiermee kun je testen hoe random een waarde werkelijk is. Tip: bijna dezelfde opdracht als 4. Zoek op hoe je random getallen genereert in C.

6. Sorteer een array van laag naar hoog, d.m.v. een bubblesort routine (link: <https://nl.wikipedia.org/wiki/Bubblesort>).

7. Schrijf een programma om het gemiddelde, de mediaan en de modus van een array te vinden.  
Tip: gebruik een redelijk grote array. En gebruik voor het gemiddelde, de mediaan en de modus, functies waar je de array aan doorgeeft (hele belangrijke voor later!).

8. Schrijf een programma dat in een array een bepaalde waarde zoekt. Doe dit op een lineaire manier (m.a.w. doorloop de array gewoon van begin tot eind, totdat je gevonden hebt wat je zocht).

9. Schrijf een programma dat in een array een bepaalde waarde zoekt. Doe dit met de methode van een binary search (<https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search>).

1. Schrijf een programma die een array van 5 elementen allemaal op 0 zet:

// Zet de elementen van een array op 0.

#include <stdio.h>

int main(void)

{

int n[5]; // n is een array van 5 integers

// zet de elementen op 0 (size\_t bij de teller geeft aan dat het een geheel getal betreft, dat gebruikt wordt).

for (size\_t i = 0; i < 5; ++i) {

n[i] = 0; // zet element op index i op 0

}

printf("%s%13s\n", "Index", "Waarde");

// zet inhoud van array in tabelformaat

for (size\_t i = 0; i < 5; ++i) {

printf("%7u%13d\n", i, n[i]);

}

}

2. Schrijf een programma die een array met een lijst initialiseert.

// Vul de elementen van een array met een (begin) lijst.

#include <stdio.h>

int main(void)

{

// gebruik een lijst met waarden om de array te vullen

int n[5] = {32, 27, 64, 18, 95};

printf("%s%13s\n", "Index", "Waarde");

// wederom tabel voor de uitvoer

for (size\_t i = 0; i < 5; ++i) {

printf("%7u%13d\n", i, n[i]);

}

}

3. Schrijf een programma die de waarden in een array bij elkaar optelt.

// De waarden in een array bij elkaar optellen.

#include <stdio.h>

#define GROOTTE 12 //de grootte van de array.

int main(void)

{

// weer een lijst met waarden om array te vullen

int a[GROOTTE] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 };

int total = 0; // som van de waarden van de array

// en hier het echte optellen

for (size\_t i = 0; i < GROOTTE; ++i) {

total += a[i]; //of: total = total + a[i];

}

printf("De som van de waarden in de array is: %d\n", total);

}

4. 40 Studenten zijn gevraagd om een enquête in te vullen om de kwaliteit van de worstenbroodjes in de Jumbo te meten. Ze konden de kwaliteit aangeven op een schaal van 1 tot 10 (1 = afschuwelijk, 10 = fantastisch). Zet de 40 uitkomsten in een integer array en geef aan hoe vaak de kwaliteit is gemeten (dus hoe vaak is 1 gekozen, hoe vaak 2, hoe vaak 3, enz.).

// Worstenbrood kwaliteit.

#include <stdio.h>

#define STUDENTEN\_AANTAL 40 // grootte van de arrays

#define HOE\_VAAK 11

//

int main(void)

{

// zet de teller die berekenen hoe vaak iets gekozen is op 0.

int teller[HOE\_VAAK] = {0}; //kortere manier dan een for-loop

// zet de (verzonnen) waardes van de studenten in de array

int antwoorden[STUDENTEN\_AANTAL] = {1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10};

// voor elk antwoord, selecteer de waarde in het element van STUDENTEN\_AANTAL en gebruik die waarde als index om te weten of je de huidige waarde in HOE\_VAAK moet verhogen.

for (size\_t antwoord = 0; antwoord < HOE\_VAAK; ++antwoord) {

++teller[antwoorden[antwoord]];

}

//regel “++teller[antwoorden[antwoord]];” had ook zo gekund:

//int index;

//index = antwoorden[antwoord];

//teller[index] = teller[index]+1;

//nu zit het in een regel (maar dat leest lastiger)

// toon resultaten

printf("%s%17s\n", "Kwaliteit", "Hoe vaak gemeten");

for (size\_t kwaliteit = 1; kwaliteit < STUDENTEN\_AANTAL; ++kwaliteit) {

printf("%6d%17d\n", kwaliteit, antwoorden[kwaliteit]);

}

}

5. Schrijf een programma die een dobbelsteen 120.000.000 keer gooit. Hou bij hoe vaak telkens een zijde is gegooid. Hiermee kun je testen hoe random een waarde werkelijk is. Tip: bijna dezelfde opdracht als 4. Zoek op hoe je random getallen genereert in C.

// Gooi een dobbelsteen 120.000.000 keer

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define ZIJDES 7 //eentje extra om op 1 te beginnen

int main(void)

{

unsigned int aantal\_keer[ZIJDES] = {0}; // leeg maken

srand(time(NULL)); // zet random nummer generator aan

// nu 120.000.000 keer gooien

for (unsigned int worp = 1; worp <= 120000000; ++worp) {

size\_t zijde = 1 + rand() % 6;

++aantal\_keer[zijde];

}

printf("%s%17s\n", "Zijde", "Aantal");

for (size\_t zijde = 1; zijde < ZIJDES; ++zijde) {

printf("%4d%17d\n", zijde, aantal\_keer[zijde]);

}

}

6. Sorteer een array van laag naar hoog, d.m.v. een bubblesort routine.

// Sorteer een array van laag naar hoog

#include <stdio.h>

#define GROOTTE 10 //grootte array

int main(void)

{

// initialize array

int a[GROOTTE] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};

puts("Array in originele vorm:");

for (size\_t i = 0; i < GROOTTE; ++i) {

printf("%4d", a[i]);

}

// bubble sort

for (unsigned int lus = 1; lus < GROOTTE; ++lus) {

// aantal vergelijkingen per lus te controleren

for (size\_t i = 0; i < GROOTTE - 1; ++i) {

// vergelijk naast elkaar liggende waarden en verwissel

// die als eerste groter is dan de tweede

if (a[i] > a[i + 1]) {

int bewaar = a[i]; //tijdelijke variabele

a[i] = a[i + 1];

a[i + 1] = bewaar;

}

}

}

puts("\nArray gesorteerd in oplopende volgorde:");

// output sorted array

for (size\_t i = 0; i < GROOTTE; ++i) {

printf("%4d", a[i]);

}

puts("");

}

7. Schrijf een programma om het gemiddelde, de mediaan en de modus van een array te vinden.  
Tip: gebruik een redelijk grote array. En gebruik voor het gemiddelde, de mediaan en de modus, functies waar je de array aan doorgeeft (hele belangrijke voor later!).

// Data analyse met een array

// Bereken gemiddelde, mediaan en modus.

#include <stdio.h>

#define GROOTTE 99

// function prototypes //dat moet hier, anders weet de compiler niet wat hij moet doen met de functies.

void gemiddelde(const unsigned int antwoord[]);

void mediaan(unsigned int antwoord[]);

void modus(unsigned int freq[], const unsigned int antwoord[]) ;

void bubbleSort(unsigned int a[]);

void printArray(const unsigned int a[]);

int main(void)

{

unsigned int aantal[10] = {0}; // initialize array aantal

// initialize array arraydata

unsigned int arraydata[GROOTTE] =

{6, 7, 8, 9, 8, 7, 8, 9, 8, 9,

7, 8, 9, 5, 9, 8, 7, 8, 7, 8,

6, 7, 8, 9, 3, 9, 8, 7, 8, 7,

7, 8, 9, 8, 9, 8, 9, 7, 8, 9,

6, 7, 8, 7, 8, 7, 9, 8, 9, 2,

7, 8, 9, 8, 9, 8, 9, 7, 5, 3,

5, 6, 7, 2, 5, 3, 9, 4, 6, 4,

7, 8, 9, 6, 8, 7, 8, 9, 7, 8,

7, 4, 4, 2, 5, 3, 8, 7, 5, 6,

4, 5, 6, 1, 6, 5, 7, 8, 7};

// alles uitrekenen

gemiddelde(arraydata);

mediaan(arraydata);

modus(aantal, arraydata);

}

// bereken gemiddelde van de array arraydata

void gemiddelde(const unsigned int antwoord[])

{

printf("%s\n%s\n%s\n", "\*\*\*\*\*\*\*\*", " Gemiddelde", "\*\*\*\*\*\*\*\*");

unsigned int totaal = 0; // variabele voor de som van het totaal

// alles optellen

for (size\_t j = 0; j < GROOTTE; ++j) {

totaal += antwoord[j];

}

printf("Het gemiddelde is gelijk aan\n"

"alle data items gedeeld door het aantal data items (%u). Het gemiddelde voor\n"

"deze cijfers is: %u / %u = %.4f\n\n",

GROOTTE, totaal, GROOTTE, (double) totaal / GROOTTE);

}

// Mediaan berekenen

void mediaan(unsigned int antwoord[])

{

printf("\n%s\n%s\n%s\n%s",

"\*\*\*\*\*\*\*\*", " Mediaan", "\*\*\*\*\*\*\*\*",

"De ongesorteerde data is:");

printArray(antwoord);

bubbleSort(antwoord); // sorteer de array

printf("%s", "\n\nDe gesorteerde array is");

printArray(antwoord);

// toon mediaan

printf("\n\nDe mediaan item ligt op positie %u \n"

"van een array met %u gesorteerde elementen.\n"

"Voor deze data is de mediaan %u.\n\n",

GROOTTE / 2, GROOTTE, antwoord[GROOTTE / 2]);

}

// Modus (meest voorkomende getal)

void modus(unsigned int freq[], const unsigned int antwoord[])

{

printf("\n%s\n%s\n%s\n", "\*\*\*\*\*\*\*\*", " Modus", "\*\*\*\*\*\*\*\*");

for (size\_t rating = 1; rating <= 9; ++rating) {

freq[rating] = 0;

}

// frequenties berekenen

for (size\_t j = 0; j < GROOTTE; ++j) {

++freq[antwoord[j]];

}

// mooi wegzetten

printf("%s%11s%19s\n\n%54s\n%54s\n%54s\n\n",

"Arraydata", "Aantal", "Histogram",

"1 1 2 2", "5 0 5 0 5", "V V V V V");

// output results

unsigned int grootste = 0;

unsigned int modusWaarde = 0;

for (size\_t rating = 1; rating <= 9; ++rating) {

printf("%8u%11u ", rating, freq[rating]);

// kijk welk getal hoe vaak voorkomt

if (freq[rating] > grootste) {

grootste = freq[rating];

modusWaarde = rating;

}

// Histogram, om het mooi te maken

for (unsigned int h = 1; h <= freq[rating]; ++h) {

printf("%s", "\*");

}

puts(""); // nieuwe regel

}

// toon modus

printf("\nModus is de meest voorkomende waarde.\n"

"Voor deze data is %u het vaakst voorgekomen"

" met %u keer.\n", modusWaarde, grootste);

}

// De bubblesort

void bubbleSort(unsigned int a[])

{

for (unsigned int lus = 1; lus < GROOTTE; ++lus) {

for (size\_t j = 0; j < GROOTTE - 1; ++j) {

// verwissel elementen als het nodig is

if (a[j] > a[j + 1]) {

unsigned int bewaar = a[j];

a[j] = a[j + 1];

a[j + 1] = bewaar;

}

}

}

}

// output array

void printArray(const unsigned int a[])

{

for (size\_t j = 0; j < GROOTTE; ++j) {

if (j % 20 == 0) { //nieuwe regel na 20 waarden

puts("");

}

printf("%2u", a[j]);

}

}

8. Schrijf een programma dat in een array een bepaalde waarde zoekt. Doe dit op een lineaire manier (m.a.w. doorloop de array gewoon van begin tot eind, totdat je gevonden hebt wat je zocht).

// Lineair doorlopen van een array.

#include <stdio.h>

#define GROOTTE 100

// function prototype

size\_t lineairZoeken(const int array[], int key, size\_t size);

int main(void)

{

int a[GROOTTE]; // maak array a (slechte naam!)

// maak wat data

for (size\_t x = 0; x < GROOTTE; ++x) {

a[x] = 2 \* x;

}

printf("Voer geheel getal in om te vinden: ");

int zoekSleutel; // waarde die we willen zoeken

scanf("%d", &zoekSleutel);

// probeer waarde te vinden

size\_t index = lineairZoeken(a, zoekSleutel, GROOTTE);

// toon resultaat

if (index != -1) {

printf("Waarde gevonden op index: %d\n", index);

}

else {

puts("Waarde niet gevonden :(");

}

}

// vergelijk elke index van array met zoekSleutel totdat die gevonden is, of totdat einde van array is bereikt. Geeft index terug als gevonden, anders -1 voor niet gevonden.

size\_t lineairZoeken(const int array[], int key, size\_t size)

{

// door array heen lopen

for (size\_t n = 0; n < size; ++n) {

if (array[n] == key) {

return n; // teruggeven van locatie

}

}

return -1; // en anders niks gevonden

}

9. Schrijf een programma dat in een array een bepaalde waarde zoekt. Doe dit met de methode van een binary search.

// Binary search van een gesorteerde(!) array

#include <stdio.h>

#define GROOTTE 15

// function prototypes

size\_t binarySearch(const int b[], int zoekSleutel, size\_t laag, size\_t hoog);

void printKop(void);

void printRij(const int b[], size\_t laag, size\_t mid, size\_t hoog);

int main(void)

{

int a[GROOTTE]; // maak array aan

// maak data

for (size\_t i = 0; i < GROOTTE; ++i) {

a[i] = 2 \* i;

}

printf("%s", "Voer een nummer tussen 0 en 28 in: ");

int sleutel; // hier zijn we naar op zoek

scanf("%d", &sleutel); //& = adrespointer. Komt nog.

printKop();

// zoek naar de sleutel

size\_t result = binarySearch(a, sleutel, 0, GROOTTE - 1);

// toon resultaten

if (result != -1) {

printf("\n%d gevonden op index/plaats %d\n", sleutel, result);

}

else {

printf("\n%d niet gevonden\n", sleutel);

}

}

// function om de binaire zoekmethode uit te voeren

size\_t binarySearch(const int b[], int zoekSleutel, size\_t laag, size\_t hoog)

{

// loop totdat laag index groter is dan hoog index

while (laag <= hoog) {

// middelste index van subarray bepalen

size\_t middelste = (laag + hoog) / 2;

// toon subarray in deze lus

printRij(b, laag, middelste, hoog);

// als zoekSleutel gelijk is aan middelste, dan return middelste

if (zoekSleutel == b[middelste]) {

return middelste;

}

// als zoekSleutel kleiner dan middelste, zet hoog opnieuw

else if (zoekSleutel < b[middelste]) {

hoog = middelste - 1; // laag kant van array zoeken

}

// als zoekSleutel groter dan middelste, dan nieuwe laag

else {

laag = middelste + 1; // zoek hoog einde van array

}

} // end while

return -1; // zoekSleutel niet gevonden

}

// Print hoofd/kop

void printKop(void)

{

puts("\nIndexen:");

// kolom

for (unsigned int i = 0; i < GROOTTE; ++i) {

printf("%3u ", i);

}

puts(""); // nieuwe regel

// regel met - printen

for (unsigned int i = 1; i <= 4 \* GROOTTE; ++i) {

printf("%s", "-");

}

puts(""); // nieuwe regel

}

// Een regel afdrukken met he stuk van de array

// die bekeken (tussen laag-hoog) bekeken wordt

void printRij(const int b[], size\_t laag, size\_t middelste, size\_t hoog)

{

// door hele array lopen

for (size\_t i = 0; i < GROOTTE; ++i) {

// toon spaties als buiten subarray ligt

if (i < laag || i > hoog) {

printf("%s", " ");

}

else if (i == middelste) { // toon middelste

printf("%3d\*", b[i]);

}

else { // en de andere indexen van de subarray

printf("%3d ", b[i]);

}

}

puts("");

}