CPROUC/2021-2022

Jesse op den Brouw

CPROUC

Arrays

DE HAAGSE HOGESCHOOL

Array

- Een array is een samengesteld datatype.
- Met de definitie:

```
int ary[5];
```

  definiëren we een array van 5 elementen.
- We hebben nu de beschikking over 5 variabelen onder een gemeenschappelijke naam:

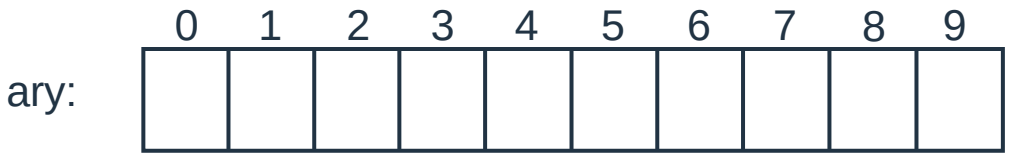
```
ary[0] ary[1] ary[2] ary[3] ary[4]
```

- Het elementnummer begint altijd bij 0 en loopt (hier) t/m 4.
- Het elementnummer is een integer.

Array – voorstellen

- We kunnen de array voorstellen als een rij hokjes:

```
int ary[10];
```



- Bij het oplossen van vraagstukken is het handig om de array op papier voor te stellen.

Array – initialisatie

- We kunnen de array als volgt initialiseren:

```
int ary[5];
ary[0] = 2;
ary[1] = 3;
ary[2] = 7;
ary[3] = 1;
ary[4] = 8;
```

- Maar sneller is directe initialisatie:

```
int ary[5] = { 2, 3, 7, 1, 8 };
```

Array – initialisatie

- Bij directe initialisatie mag het aantal elementen weggelaten worden:

```
int ary[] = { 2, 3, 7, 1, 8 };
```

- Bij directe initialisatie mag het aantal elementen opgegeven worden en mag de lijst kleiner zijn dan het aantal elementen.

```
int ary[10] = { 2, 3, 4 };
```

- De overige elementen worden met 0 geïnitialiseerd.

Array – grenzen

- Let op de grenzen van de array:

```
int ary[5];  /* elementnummer 0 t/m 4 */

ary[-1] = 2; /* ligt buiten de array */
ary[5] = 3;  /* ligt buiten de array */
```

- De C-compiler voegt geen code toe om de grenzen te bewaken. Dit moet door de programma gerealiseerd worden.

Array – som bepalen

- Met behulp van een for-herhaling kunnen we “langs de array lopen”.

```
int ary[5] = { 2, 3, 7, 1, 8 };
int i, som = 0;

for (i = 0; i < 5; i = i + 1)
{
    som = som + ary[i]; /* i is het elementnummer */
}
printf("De som is: %d\n", som);
```

Array – aantal elementen uitrekenen

- Het is mogelijk om het aantal elementen van een array tijdens *compile-time* te berekenen.

```
int ary[] = { 2, 3, 7, 1, 8 };

int nr_elem = sizeof ary / sizeof ary[0];
```

- `sizeof` geeft het aantal *bytes* van het argument. Hier wordt dus de arraygrootte in bytes gedeeld door de grootte van één element in bytes.

Array – som bepalen

- We kunnen nu de array gemakkelijk aanpassen.

```
int ary[] = { 2, 3, 7, 1, 8 };
int i, som = 0;
int nr_elem = sizeof ary / sizeof ary[0];

for (i = 0; i < nr_elem; i = i + 1)
{
   som = som + ary[i]; /* i is het elementnummer */
}
printf("De som is: %d\n", som);
```

Array – inlezen array

- Een array kan worden ingelezen:

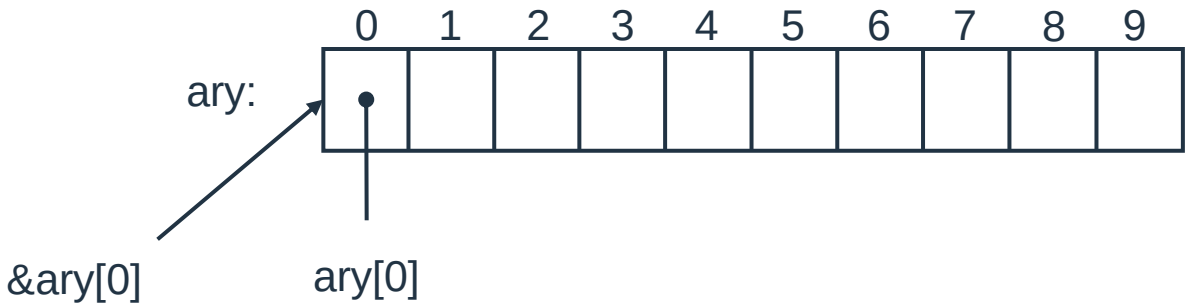
```
int ary[5] = { 0 }; /* alle elementen 0 */
int i;

for (i = 0; i < 5; i = i + 1)
{
   printf("Geef getal %d: ", i);
   scanf("%d", &ary[i]); /* let op de & */
}
```

Array – functies

- Een array kan dienen als een argument/parameter van een functie.
- Een array kan *niet* als returnwaarde dienen.
- Dat komt door de wijze van argument/parameter-overdracht.
- Bij een array-argument wordt niet de hele array overgedragen, maar het *adres van het eerste element*.
- Dat is veel efficiënter dan het de hele array overdragen.
- Het gevolg is dat een functie niet “weet” hoe groot een meegegeven array is.
- Er is dus een extra argument/parameter nodig met het aantal elementen van de array.

Array – functies

- Bij een array-argument wordt niet de hele array overgedragen, maar het *adres van het eerste element*.



- ary[0] → inhoud van element 0
- &ary[0] → adres van element 0

DE HAAGSE HOGESCHOOL

Array – functie berekenen som

- Een functie voor het berekenen van de som van de elementen in een array.

```
int berekensom(int r[], int grootte) /* r[] -> array */
{
   int som = 0;
   for (i = 0; i < grootte; i = i + 1)
   {
      som = som + r[i]; /* i is het elementnummer */
   }
   return som;
}
```

Array – aanroepen functie

- We kunnen nu de functie aanroepen.

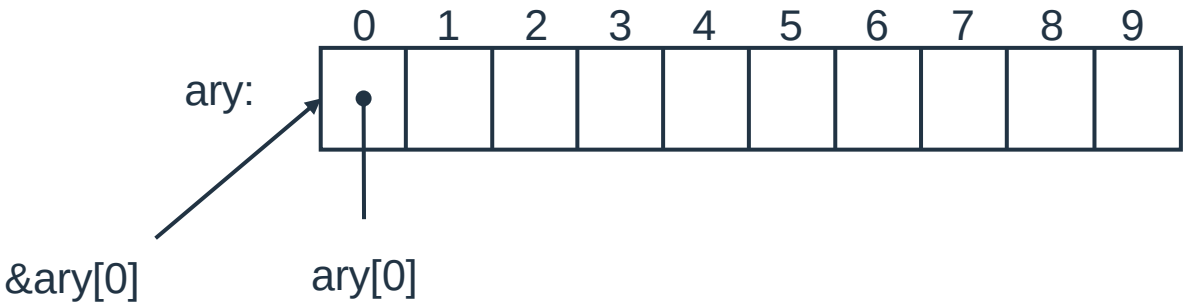
```
int ary[] = { 2, 3, 7, 1, 8 };

int nr_elem = sizeof ary / sizeof ary[0];
int som;

som = berekensom(ary, nr_elem);
```

- Gebruik naam van array als argument.

Array – aanpassen in de functie

- We kunnen de array in de functie aanpassen.
- Er wordt immers het beginadres van de array meegegeven.
- We weten dus waar de array (de parameter in de functie) ligt opgeslagen.

```
som = berekensom(ary, nr_elem);
```



DE HAAGSE HOGESCHOOL

Array – aanpassen in de functie

```
int patcharray(int ary[], int siz, int what, int with) {
   int count = 0;

   for (int pos = 0; pos < siz; pos = pos + 1) {
      if (ary[pos] == what) { /* found? */
          ary[pos] = with;    /* patch! */
          count = count + 1;
      }
   }

   return count;
}
```

Twee-dimensionale array

- Met de definitie:

```
int ary[5][3];
```

  definiëren we een twee-dimensionale array van 5x3 elementen.
- Het eerste elementnummer noemen de rij (row).
- Het tweede nummer noemen we de kolom (column).
- We kunnen nu een element selecteren:

`ary[0][0]`, `ary[1][2]`, `ary[2][1]`

- Let erop dat `ary[1][2]` en `ary[2][1]` verschillende elementen zijn.

Twee-dimensionale array

- Directe initialisatie met dubbele accolades:

```
int ary[5][3] = { { 1, 2, 3}, {4, 7, 2}, {9, 4, 6},
                  { 0, 5, 8}, {8, 1, 9} };
```

- Aantal rijen en kolommen bepalen:

```
int rows = sizeof ary / sizeof ary[0];
int cols = sizeof ary[0] / sizeof ary[0][0];
```

Twee-dimensionale array – som van

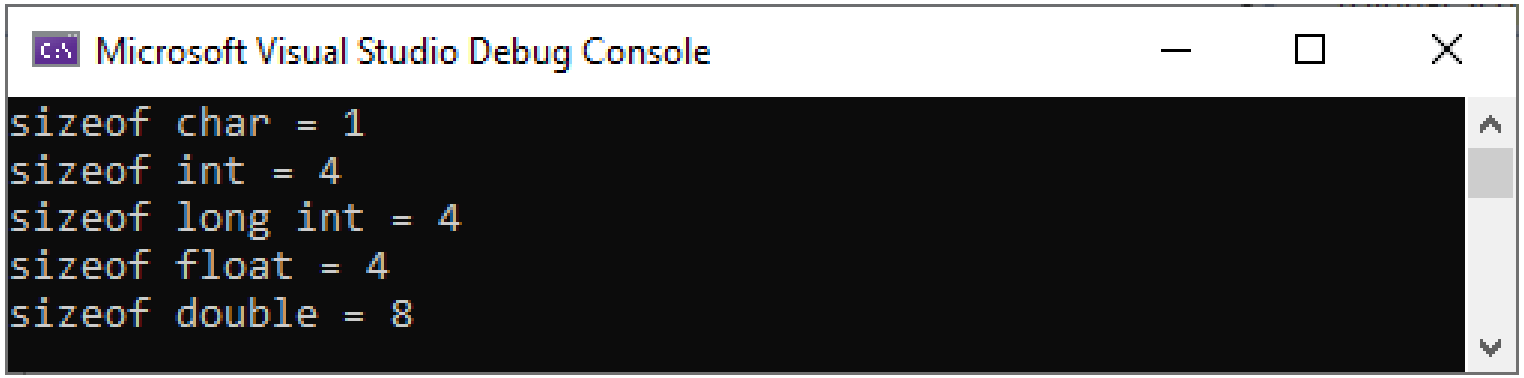
- Som bepalen van de tweedimensionale array:

```
int rows = sizeof ary / sizeof ary[0];
int cols = sizeof ary[0] / sizeof ary[0][0];

for (int r = 0; r < rows; r = r + 1) {
   for (int c = 0; c < cols; c = c + 1) {
      som = som + ary[r][c];
   }
}
```

Sizeof revisited

- We kunnen `sizeof` ook gebruiken voor datatypes:

```
printf("sizeof char = %d\n", sizeof (char));
printf("sizeof int = %d\n", sizeof (int));
printf("sizeof long int = %d\n", sizeof (long int));
printf("sizeof float = %d\n", sizeof (float));
printf("sizeof double = %d\n", sizeof (double));
```



CPROUC/2021-2022

Jesse op den Brouw

CPROUC

Strings

DE HAAGSE HOGESCHOOL

Strings

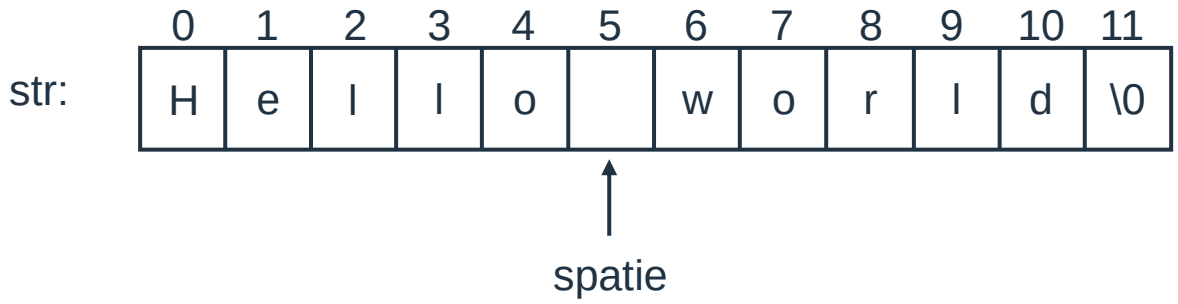
- Een string is een array van karakters afgesloten door een null-byte of null-karakter.
- Deze null-byte geeft het einde van de string aan.
- In een null-byte zijn alle bits 0.
- In C wordt dat geschreven als ‘\0’ (backslash-nul).
- Er is dus altijd één karakter extra nodig voor de ‘\0’.

Strings

- We kunnen een string bij definitie direct initialiseren, de null-byte wordt automatisch toegevoegd.

```
char str[] = "Hello world"; /* lengte automatisch bepaald */
```

- String in het geheugen:



DE HAAGSE HOGESCHOOL

Strings

- We kunnen een string afdrukken:

```
printf("%s", str); /* gebruik %s om string af te drukken */
```

- We kunnen een string inlezen:

```
scanf("%s", str); /* noot: geen &, of gebruik &str[0] */
```

- Dit gaat fout als er meer tekens worden ingevoerd dan de lengte van de string. Gebruik dan bv. `"%99s"` voor een string van 100 karakters (inclusief de null-byte).

Strings

- We kunnen de lengte van de string eenvoudig bepalen met een while-herhaling. De string eindigt immers met een null-byte.

```
int len = 0;
char str[] = "Hello world";

while (str[len] != '\0') {
    len = len + 1;
}

printf("Lengte is %d\n", len); /* levert 11 op */
```

Strings

- We kunnen de lengte van de string eenvoudig bepalen met een for-herhaling. De string eindigt immers met een null-byte.

```
int len;
char str[] = "Hello world";

for (len = 0; str[len] != '\0'; len = len + 1);

printf("Lengte is %d\n", len); /* levert 11 op */
```

Strings

- Een functie voor het bepalen van de lengte van een string is niet zo ingewikkeld:

```
int stringlength(char str[]) {
   int len;

   for (len = 0; str[len] != '\0'; len = len + 1);

   return len;
}
```

Strings

- De standard library kent een groot aantal string-functies. Voor gebruik moet het header-bestand `string.h` geladen worden.

```
strlen(str)          lengte string
strcpy(to, from)     kopiëren string
strcat(to, from)     from achter to plaatsen
```

- Let erop dat bij `strcpy` en `strcat` de ruimte van `to` groot genoeg moet zijn anders volgt een buffer overflow!

Strings

- De functie `strcmp` vergelijkt twee strings, op volgorde van de gebruikte tekenset (bijvoorbeeld ASCII).

```
int strcmp(char s1[], char s2[])
```

- Als s1 en s2 identiek zijn, wordt 0 teruggegeven.
- Als s1 “kleiner” is dan s2, wordt een negatief getal teruggegeven.
- Als s1 “groter” is dan s2, wordt een positief getal teruggegeven.

ASCII-code

Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex	
0	00	NUL	16	10	DLE	32	20		48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
1	01	SOH	17	11	DC1	33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
2	02	STX	18	12	DC2	34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
3	03	ETX	19	13	DC3	35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
4	04	EOT	20	14	DC4	36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
5	05	ENQ	21	15	NAK	37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
6	06	ACK	22	16	SYN	38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
7	07	BEL	23	17	ETB	39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
8	08	BS	24	18	CAN	40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
9	09	HT	25	19	EM	41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
10	0A	LF	26	1A	SUB	42	2A	*	58	3A	:	74	4A	J	90	5A	Z	106	6A	j	122	7A	z
11	0B	VT	27	1B	ESC	43	2B	+	59	3B	;	75	4B	K	91	5B	[	107	6B	k	123	7B	{
12	0C	FF	28	1C	FS	44	2C	,	60	3C	<	76	4C	L	92	5C	\	108	6C	l	124	7C	|
13	0D	CR	29	1D	GS	45	2D	-	61	3D	=	77	4D	M	93	5D	]	109	6D	m	125	7D	}
14	0E	SO	30	1E	RS	46	2E	.	62	3E	>	78	4E	N	94	5E	^	110	6E	n	126	7E	~
15	0F	SI	31	1F	US	47	2F	/	63	3F	?	79	4F	O	95	5F	_	111	6F	o	127	7F	DEL

Strings

- Voorbeelden

```
int t;
t = strcmp("jan", "jan");       /* levert 0 op */
t = strcmp("jan", "janus");     /* levert <0 op */
t = strcmp("kaas", "brood");    /* levert >0 op */

t = strcmp("jan", "Jan");       /* levert >0 op */
```

Strings – array van strings

- Voorbeeld van een array van strings:

```
char day[7][10] = { "zondag", "maandag", "dinsdag",
                    "woensdag", "donderdag", "vrijdag",
                    "zaterdag" };
```

- De 7 in `day[7][10]` mag weggelaten worden, want de compiler kan dat uitrekenen. De 10 mag niet weggelaten worden.

Strings – array van strings afdrukken

- Voorbeeld van een array van strings afdrukken

```
char day[7][10] = { "zondag", "maandag", "dinsdag",
                    "woensdag", "donderdag", "vrijdag",
                    "zaterdag" };

int aantal = sizeof day / sizeof day[0]; /* bereken aantal strings */

for (int i = 0; i < aantal; i = i + 1)
{
   printf("%s\n", day[i]); /* day[i] → beginadres string */
}
```

let’s change