Basisconcepten C

# Situering

* C staat dicht bij de hardware, waardoor het sneller is.

# Basisconcepten

**Commentaar:**

* /\* \*/
* // Sinds C99

**Return 0 in Main:**

* Return 0 indien programma succesvol is afgesloten.
* Kan je opvragen via %ErrorLevel%

**Karakter:** tussen enkele aanhalingstekens: ‘a’

**String:** tussen dubbele aanhalingstekens: “a”

‘a’ ≠ “a”

# Programmastructuur

1. Korte zin over de inhoud
2. Includes (Preprocessing directives) (niet verplicht bovenaan)
3. Declaraties variabelen/constanten
4. Declaraties functies
5. Main programma
6. Definities functie

# Preprocessing directives (Voorverwerkingsrichtlijnen)

* Beginnen met een #
* Bestemd voor de compiler, ze voeren deze instructies eerst uit alvorens de code te compileren

#include <stdio.h>

#define PI 3.14159f

# Variabelen declareren

* Kan beginnen met een underscore of letter, maar geen letter.
* geen hoofdletters

# Fundamentele datatypes in C

Alle fundamentele types zijn numeriek, ook char.

Lengte van het type is machine-afhankelijk. Lengte wordt uitgedrukt in bytes.

**Reële types**

sizeof(float) ≤ sizeof(double) ≤ sizeof(long double)

* Float: 4 byte 6 decimalen
* Double: 8 byte 15 decimalen
* Long Double: 10 byte 19 decimalen

Let op: dit zijn de standaarden. Kunnen nog altijd verschillen van machine tot machine.

1. # include <stdio.h>
2. # include <stdbool.h>
3. int main() {
4. printf("%lu \n", sizeof(char)); // 1 byte
5. printf("%lu \n", sizeof(unsigned char)); // 1 byte
6. printf("%lu \n", sizeof(signed char)); // 1 byte
7. printf("%lu \n", sizeof(short)); // 2 bytes
8. printf("%lu \n", sizeof(short int)); // 2 bytes
9. printf("%lu \n", sizeof(signed short)); // 2 bytes
10. printf("%lu \n", sizeof(signed short int)); // 2 bytes
11. printf("%lu \n", sizeof(unsigned short)); // 2 bytes
12. printf("%lu \n", sizeof(unsigned short int)); // 2 bytes
13. printf("%lu \n", sizeof(int)); // 4 bytes
14. printf("%lu \n", sizeof(signed)); // 4 bytes
15. printf("%lu \n", sizeof(signed int)); // 4 bytes
16. printf("%lu \n", sizeof(unsigned int)); // 4 bytes
17. printf("%lu \n", sizeof(unsigned)); // 4 bytes
18. printf("%lu \n", sizeof(long)); // 8 bytes
19. printf("%lu \n", sizeof(long int)); // 8 bytes
20. printf("%lu \n", sizeof(signed long)); // 8 bytes
21. printf("%lu \n", sizeof(signed long int)); // 8 bytes
22. printf("%lu \n", sizeof(unsigned long)); // 8 bytes
23. printf("%lu \n", sizeof(unsigned long int)); // 8 bytes
24. printf("%lu \n", sizeof(long long)); // 8 bytes
25. printf("%lu \n", sizeof(long long int)); // 8 bytes
26. printf("%lu \n", sizeof(signed long long)); // 8 bytes
27. printf("%lu \n", sizeof(signed long long int)); // 8 bytes
28. printf("%lu \n", sizeof(unsigned long long)); // 8 bytes
29. printf("%lu \n", sizeof(unsigned long long int)); // 8 bytes
30. printf("%lu \n", sizeof(\_Bool)); // 1 byte
31. printf("%lu \n", sizeof(bool)); // 1 byte
32. return 0;
33. }

# Literals

{prefix}opt {+|-}opt getal {suffix}opt

**Prefix:**

* Decimaal Geen
* Octaal 0
* Hexadecimaal 0x of 0X

**Suffix:**

* int niets
* int unsigned u of U
* long l of L
* unsigned long ul of UL
* long long ll of LL
* unsigned long long ull of ULL
* double niets
* float f of F
* long double l of L

**Voorbeelden:**

* int i = 0x200 Dus hexadecimaal getal => 512 in decimale vorm

# Constanten

* const type NAAM = waarde; C99
* #define NAAM waarde

**Vragen:**

* Welke standaard gebruiken wij nu in de lessen? Is dit C99, C11 of nog de oudere variant? Of moeten we zorgen dat alle oudere versies ook compatibel zijn en daar zorgt de pedantic voor? Want mogen we de // commentaar gebruiken ?
* const float Pi = 3.14159f; Die f is volgens mij optioneel?

# Impleciete conversies

1. **float** x = 2.8; // 2.800000
2. **int** y = x; // 2 impliciet
3. **int** z = (**int**) x; // 2 expliciet
4. **double** a = 5 / 3; // 1.000000
5. **double** b = (**double**) 5 / 3; // 1.666667

Gebruik casts om aan te geven wat de bedoeling is. Steun niet te veel op impliciete conversieregels!

# Operatoren

* int a = 0;
* int b = 0;
* int c = 0;
* int d = 0;
* b = a++; // a = 1; b = 0;
* d = ++c; // c = 1; d = 1;

**Slide 23 eens bekijken wat niet mag met in/decrement operatoren**

# Printf

**%[-][w][.p]c**

* Linkse alignatie -
* Minimale breedte w (width)
* Maximaal karakters of cijfers na de komma p (precisie)
* Conversiekarakter c
  + (l)d decimaal geheel getal
  + (l)o octaal geheel getal
  + (l)x hexadecimaal geheel getal
  + (l)f reëel getal
  + e reëel getal in wetenschappelijke notatie
  + c karakter

# Scanf

* %d int
* %f float
* %c char
* %lf double
* %Lf long double
* %ld long
* %lld long long
* %hd short
* %s s

Resultaat van scanf: aantal succesvol gelezen variabelen

# getchar / putchar

* c = getchar(); // gets a character (an unsigned char)
* putchar(c); // writes a character to stdout

# Functies

Functies moeten gedeclareerd worden **voor** gebruik.

**Effect functie-oproep:**

* Argumenten worden geëvalueerd
* Waarde van elk argument wordt toegekend aan formele parameter van functiedefinitie
* Controle wordt overgedragen aan functie
* Resultaatwaarde vervangt functie-oproep in de oproepende context

**Call by value:**

waarde wordt doorgegeven, een copy indien het niet om een adres gaat.

# Recursieve functies

**(Direct) recursief**:

Een functie heet (direct) recursief als het zichzelf aanroept.

**Indirect recursief:**

Een functie heet indirect recursief als de aanroep van zichzelf via een of meerdere andere functies gaat.

**Recursieve oplossing bevat:**

1. Een of meer basisgevallen
2. Een of meer recursieve gevallen

# Arrays

type naam[grootte\_array]; // Grootte\_array moet een constante zijn. (#define …)

**OutOfBounds:**

Elementen uitschrijven met een index buiten bereik => geen foutmelding

**Grootte array:**

sizeof(naam)/sizeof(type)

**Opmerkingen:**

* De grootte van een array moet steeds expleciet meegegeven worden aan de functie
* Het resultaat van een functie kan geen array zijn.
* Het adres van de array wordt meegegeven als argument

**Toepassingen:**

* Horner
* Lineair zoeken in een ongesorteerde array
* Array opvullen in volgorde