Notities les 3

Jelle De Bock

October 6, 2015

1 Bewerkingen op pointers

Korte recap van vorige les.

```
void wissel(int a, int b){
 int hulp =a;
 a = b;
 b = hulp;
}

void wissel(int * a, int * b){
 int hulp = *a;
 *a = *b;
 *b = hulp;
}



int main()
{
  int g1=8, g2=10;
 //deze functie geeft enkel de waardes van g1 en g2
 //door procedure wijzigt niets
 wissel(g1,g2);
 //nu wordt het adres naar de functie doorgegeven,
 //de functie wissel moet nu ook lichtjes gewijzigd worden
 wissel(&g1,&g2);

 }
```

Opmerking Wanneer je belooft om niets aan de array te veranderen, dan zet je er const voor...
Is vooral belangrijk om aan te geven aan iemand die de code leest dat de array

niet wijzigt.
Bijvoorbeeld...

```
void bepaal_min_max(const int *t, int n, int * min, int *max)
{
    int i=0;
    *min = t[0];      //gelijk aan *t
    *max = t[0];      //idem

    for(i=1;i<n;i++)
    {
        if(*min<t[i])    //gelijk aan *(t+i)
            //...
    }
    //int[] is array pointer (kent bijvoorbeeld zijn size)
    //int * is een pointer naar een integer (geen size)
}
```

Oefening kijk of een array omkeerbaar is. Je kan hem spiegelen

```
int is_omkeerbaar(const int * array, int n)
{
  const int *b = array;     //array is ook const
  const int *e = array+n-1; //array is const

  //!= mag niet want pointers kunnen elkaar "voorbijsteken"
  while(b<e)
  {
    if(*b!=*e)
        return 0;
    e--;
    b++;
  }
  return 1;
}
```

Opmerking wanneer je twee pointers van elkaar aftrekt, dan kijk je hoe ver deze uit elkaar staan.
Pointer omzetten naar int ((int)pointer) geeft je een waarde. In principe zit hierin een hexadecimaal getal. Deze is Verschil (geredeneerd) * aantal bytes per gegevenstype. Voorbeeld 4 * 8 bytes (integer) = 32.

2 Constante pointer

Wanneer je een constante definieert dan moet je hem onmiddelijk een waarde toekennen.

```
int * const q = &n;
//hier is q constant, constante pointer (adres niet wijzigbaar)
```

3 C-string

Een string in C is een pointer naar een sliert van karakters, afgesloten met een null karakter.

3.1 Het null karakter

Het karakter kan voorgesteld worden door \0 of 0.

3.2 Manieren om strings te declareren

- Met een pointer

```
char *s1 = "string";
//je krijgt nu |'s'|'t'|'r'|'i'|'n'|'g'|'\0'|
//je kan niet inlezen in deze string
```

- Met een array(pointer)

```
char s2[80];
//je kan nu inlezen (max 79 chars)
scanf("%s",s2);
//In principe moet je ervoor zorgen dat de gebruiker maar max
//79 chars kan invoeren door
scanf("%79s",s2);
//Als je meer inleest dan blijft de overschot staan en worden
//deze bij volgende inputs gebruikt.
```

- Lichte variant

```
char s3[]="voorbeeld";
//nu beperk je uzelf tot strings van max 9 characters
```

```
char * fgets(char *s, int n; stream)
fgets(s2,80,stdin);
//80 inclusief! null terminator
//moet verschillend van null zijn bij success
```

Nadeel: Leest een volledige lijn in. Leest tot en met de enter of maximaal het tweede argument - 1. Hij stopt van zodra hij een enter vindt. Dus 1 en dan enter zorgt bijvoorbeeld voor problemen.
Tip: kan wel eens handig zijn bij het "flushen" van de buffer.

3.3 Enkele handige functies

Op examen is een API voor handen, hieronder enkele opmerkingen bij deze string functies

- *strlen* geeft lengte zonder nullcharachter
- *strcpy, strcat* moet je ervoor zorgen dat er voldoende ruimte is.
- om de inhoud van strings te vergelijken gebruik je strcmp! niet ==
- strcat gaat achter de destination de source plaatsen.

3.4 Enkele oefeningen

Custom lengte functie

```
int strlen(const char *s)
{
    int i = 0;
    while(s[i])
        i++;
    return i;
}
//lichtjes anders dan oplossing op slides
```

Custom kopieer functie

```
void strcopy(const char *src, char *dest)
{
    while(*s)
    {
      *d=*s;
      d++;
      s++;
    }
    *d=0;
}
```

Zet om: hoofdletters naar kleine letters

```
void zetom(char * s)
{
    while(*s)
    {
        if(*s>='A' && *s<='Z')
            *s = 'a'+(*s-'A');
        s++;
    }
}
//Wisjedatje: int i = ch-'\0' geeft je de
//int waarde van je char
```

4 argc en argv

Parameters dat je kan meegeven vanop de cmd line. Dit moet je toevoegen aan uw main.

```
int main(int argc, char ** argv)
{
    //argc = het aantal argumenten
    //sowieso 1 als je geen extra strings meegegeven hebt.
    //argv = char* (* argv->de string) en char* array
    //dus argv is een array van strings (char *)
}
```

5 Pointers als functieresultaat

Geeft in plaats van een raw value een adres waar de waarde zich bevindt terug.

Opmerking bij oefening eerste_hoofdletters Je krijgt een warning omdat je een gewone pointer wil teruggeven en je een consante als argumen krijgt. De const wordt in zulke gevallen meestal weggelaten omdat je de gebruiker niet wil lastig vallen met deze problematiek.

6 Pointer naar functies

Je kan op dynamische wijze kiezen voor welke data je welke functie gebruikt. Zie slides voor uitgewerkt voorbeeld.

7 Structures

Stel je wil een punt definieren:

```
typedef struct{
    double x,y;
} punt;
//hier staat eig. ik ga een struct definieren en noem hem 'punt'
punt p1,p2;
//Initialisatie:net als bij arrays...
punt p1 = {3.5,3.6};
p1=0; //alles wordt 0
```

Wistjedatjes

- Alle velden van een struct zijn public.
- Structs aan elkaar gelijkstellen, kopieert hetgeen van de eerste naar de tweede
- Vergelijken en dergelijke moet je veld per veld gaan doen...
- Je kan structs in structs steken