typedef

Döper om ”besvärliga typer”

```
typedef unsigned long int ulong;
ulong ul;
typedef char  *text;
typedef float falt[10];
typedef float (*funk_pek)(int);

text t = "Hej";
falt a;
funk_pek pf;
funk_pek apf[10];
```

Deklaration av `struct`

```
struct person_struct
{
  char fornamn[20];
  char efternamn[30];
  int fodd_ar;
  int gift;
};

struct person_struct lisa, oscar;  // variabeldeklarationer
```

Elegantare:

```
typedef struct person_struct
{
  char fornamn[20];
  char efternamn[30];
  int fodd_ar;
  int gift;
} persontyp;

persontyp lisa, oscar;                // variabeldeklarationer
```

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

int main()
{
  persontyp e;
  strcpy(e.fornamn,"Kalle");
  strcpy(e.efternamn,"Nilson");
  e.fodd_ar = 1996;
  e.singel = 1;
  printf("%s, född år %i, är ", e.fornamn, e.fodd_ar);
  if (!e. gift)
    printf("inte");
  printf("gift \n");
}
```

```
void print_person1(persontyp pe)
{
  printf("%s %s\n", pe.fornamn, pe.efternamn);
  printf("Född år: %i, Civilstånd: %sgift\n",
         pe.fodd_ar, (pe.gift)?"":"o");
}

persontyp read_person1(void)
{
  persontyp pe;
  printf("Förnamn? ");
  scanf("%s", pe.fornamn);
  printf("Efternamn? ");
  scanf("%s", pe.efternamn);
  printf("Födelseår? ");
  scanf("%i", &pe.fodd_ar);
  printf("Gift (ja/nej)? ");
  char svar[4];
  scanf("%s", svar);
  pe.gift = svar[0] == 'j' || svar[0] == 'J';
  return pe;
}
```

Anrop:

```
persontyp e;
e = read_person1();
print_person1(e);
```

⇨ Kopiering av argument och returvärde!

Pekare som parametrar

```
void print_person(const persontyp *p)
{
  printf("%s %s\n", p->fornamn, p->efternamn);
  printf("Född år: %i, Civilstånd: %sgift\n",
         p->fodd_ar, (p->gift)?"":"o");
}


void read_person(persontyp *p)
{
  printf("Förnamn? ");
  scanf("%s", p->fornamn);
  printf("Efternamn? ");
  scanf("%s", p->efternamn);
  printf("Födelseår? ");
  scanf("%i", &p->fodd_ar);
  printf("Gift (ja/nej)? ");
  char svar[4];
  scanf("%s", svar);
  p->gift = svar[0] == 'j' || svar[0] == 'J';
}
```

Som en ”klass”

```
// Filen person.h

#ifndef PERSON_H
#define PERSON_H

struct person_struct;
typedef struct person_struct *Person;

// "konstruktor"
Person new_person(const char fnamn[] , const char enamn[], int ar,
int gift);

// "destruktor"
void delete_person(Person this);

// "instansmetoder"
void print_person(Person this);
void read_person(Person this);

#endif
```

```
// Filen person.c

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "person.h"

struct person_struct      // gör implementering ”private”
{
  char fornamn[20];
  char efternamn[30];
  int fodd_ar;
  int gift;
};


static void safecpy(char *s1, const char *s2, int n)  // "private"
{
  strncpy(s1,s2,n);
  s1[n-1] = '\0';
}
```

```
static void check(Person this)                          // "private"
{
  if (!this)
  {
    fprintf(stderr,"Error: this == NULL\n");
    exit(99);
  }
}

Person new_person(const char fnamn[],const char enamn[],
                  int ar, int gift)
{
  Person this = (Person) malloc(sizeof (struct person_struct));
  if (this != NULL)
  {
    safecpy(this->fornamn,   fnamn, sizeof this->fornamn);
    safecpy(this->efternamn, enamn, sizeof this->efternamn);
    this->fodd_ar = ar;
    this->gift = gift;
  }
  return this;
}
```

```
void delete_person(Person this) {
  check(this);
  free(this);
}


void read_person(Person this)
{
  check(this);
  // som tidigare
}


void print_person(const Person this)
{
  check(this);
  // som tidigare
}
```

Testprogram:

```
#include "person.h"

int main()
{
  Person pe = new_person("Charlotta Evelina Birgitta",
                         "Olsson", 1955, 0);
  print_person(pe);
  read_person(pe);
  print_person(pe);
  delete_person(pe);
}
```

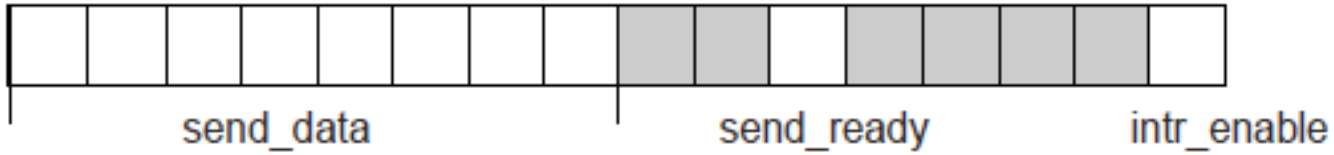
Länkade listor

```
struct elist {
   char t;
   struct elist *next;
};


struct dlist {
   char t;
   struct dlist *pred, *succ;
};
```

```
void printlist(const struct elist *p)
{
   while (p != NULL) {
      putchar(p->t);
      p = p->next;
   }
}


void printlist_rec(const struct elist *p)
{
   if (p != NULL) {
      putchar(p->t);
      printlist_rec(p->next);
   }
}
```




```
struct utregister {
   unsigned intr_enable: 1;
   unsigned            : 4;
   unsigned send_ready : 1;
   unsigned            : 2;
   unsigned send_data  : 8;
};
```

```
struct s {
   int i;
   float f;
   char c[10];
};

struct s p1;


union u {
   int i;
   float f;
   char c[10];
};

union u p2;


u.i = 289;
u.f = 156.67;  /* u.i är nu odefinierat */
u.c[0] = 'r';  /* u.f är nu odefinierat */
```

```
struct produkt {
  int nr;
  char namn[50];
  char fastpris;
  union {
     long konstant;
     long (*dagspris)(const produkt *);
  } pris;
};


/* funktion som skriver ut namn och pris på en produkt */
void printprodukt(const produkt * vp) {
  printf("Produkt %s kostar: ", vp->namn);
  if (vp->fastpris) {
     printf("%ld", vp->pris.konstant);
  }
  else {
     printf("%ld", vp->pris.dagspris(vp));
  }
}
```