CHALMERS
UNIVERSITY OF TECHNOLOGY

Maskinorienterad Programmering

C-Programming part 3

Erik Sintorn
erik.sintorn@chalmers.se

Original slides by Pedro Trancoso and Ulf Assarsson

Innehåll

- Structs, pekare till structs, arrayer av structs
- Portaddressering med structar
- Funtionspekare
- Structar med funktionspekare

Förra C lektionen

- Pekare
- Absolut addresering (portar)
- typedef, volatile, #define
- Arrays av pekare, arrays av arrays

Struct – Komposit datatyp

```
int main()
{
    // Define monter0
    int monster0_health = 5;
    int monster0_strength = 4;
    int monster0_dexterity = 3;
    int monster0_intelligence = 5;
    int monster0_weapon = 2;
    // ...

    AttackMonster(  monster0_health, monster0_strength,
                    monster0_dexterity, monster0_intelligence,
                    monster0_weapon, ...)
};
```

```
typedef struct
{
    int health, strength, dexterity;
    int intelligence, weapon;
} Monster;

int main()
{
    // Define monter0
    Monster monster0 = { 5, 4, 3, 5, 2 };
    AttackMonster( monster0 );
};
```

C Struct vs Java Class

- En Java class kan också användas för att sätta ihop en komposit typ
- Där slutar likheterna.
- C struct:
  - Inga metoder (~~`monster.attack()`~~)
  - Inga arv
  - Inga privata / publica variabler

Deklaration av struct

```
int main()
{
    struct Player
    {
        int health;
        int strength;
    };

    struct Player player_1, player_2;

    player1.health = 1;
};
```

```
int main()
{
    struct Player
    {
        int health;
        int strength;
    } player_1, player_2;

    player_1.health = 1;
};
```

```
int main()
{
    struct
    {
        int health;
        int strength;
    } player_1, player_2;

    player_1.health = 1;
};
```

Deklaration av struct

```
int main()
{
    struct Player
    {
        int health;
        int strength;
    };

    struct Player player_1, player_2;

    player1.health = 1;
};
```

```
int main()
{
    typedef struct
    {
        int health;
        int strength;
    } Player;

    Player player_1, player_2;

    player_1.health = 1;
};
```

Assignment till struct

```
int main()
{
    typedef struct
    {
        int health;
        int strength;
    } Player;

    Player player_1;
    player_1.health = 1;
    player_1.strength = 2;

};
```

```
int main()
{
    typedef struct
    {
        int health;
        int strength;
    } Player;

    Player player_1 = {1, 2};
};
```

```
int main()
{
    typedef struct
    {
        int health;
        int strength;
    } Player;

    Player player_1 = {.strength = 2, .health = 1};
};
```

Structar med structar i...

```
typedef struct { float x, y; } Position;
typedef struct
{
    int health, strength;
    Position pos;
} Player;

int main()
{
    // Initialize player
    Player player_1 = { 1, 2, { 0.0f, -2.0f }};
    // Move player
    player_1.pos.x += 5.0f;
};
```

Pekare till struct

```c
typedef struct
{
    int health, strength;
} Player;

void KillPlayer(Player * player)
{
    (*player).strength = 0;
    (*player).health = 0;
}

int main()
{
    // Initialize player
    Player player_1 = { 1, 2 };
    // Kill Player
    KillPlayer(&player_1);
};
```

```c
typedef struct
{
    int health, strength;
} Player;

void KillPlayer(Player * player)
{
    player->strength = 0;
    player->health = 0;
}

int main()
{
    // Initialize player
    Player player_1 = { 1, 2 };
    // Kill Player
    KillPlayer(&player_1);
};
```

Arrays of structs

```
typedef struct {
    char* name;
    float credits;
} Student;

Student students[] = {{"Per", 200.0f}, {"Tor", 200.0f}, {"Ulf",  20.0f}};

int main()
{
    printf("%s, %s, %s\n", students[0].name, students[1].name, students[2].name);
    return 0;
}
```

Port addressing med structs

Doomsday Device Port Specification

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	STATUS																															
4	ROCKET1																ROCKET2															
																	reserved															

```
#define DDD_PORT_BASE 0xA0001000
#define DDD_PORT_STATUS   ((volatile unsigned int *)  (DDD_PORT_BASE))
#define DDD_PORT_ROCKET1  ((volatile unsigned char *) (DDD_PORT_BASE+0x7))
#define DDD_PORT_ROCKET2  ((volatile unsigned char *) (DDD_PORT_BASE+0x6))

int main()
{
    // Arm first rocket
    *DDD_PORT_ROCKET1 = 0x33;
}
```

Port addressing med structs

Doomsday Device Port Specification

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	STATUS																															
4	ROCKET1								ROCKET2								reserved															

```c
typedef struct {
    unsigned int status;
    unsigned short unused;
    unsigned char rocket2;
    unsigned char rocket1;
} DDD;

int main()
{
    DDD * doomsday_device_1 = (volatile DDD *)0xA0001000;
    // Arm first rocket
    doomsday_device_1->rocket1 = 0x33;
}
```

Portaddressering med structs (riktigt exempel)

```
typedef struct tag_usart
{
    volatile unsigned short sr;
    volatile unsigned short Unused0;
    volatile unsigned short dr;
    volatile unsigned short Unused1;
    volatile unsigned short brr;
    volatile unsigned short Unused2;
    volatile unsigned short cr1;
    volatile unsigned short Unused3;
    volatile unsigned short cr2;
    volatile unsigned short Unused4;
    volatile unsigned short cr3;
    volatile unsigned short Unused5;
    volatile unsigned short gtpr;
} USART;
#define USART1 ((USART *) 0x40011000)
```

Example:

```
while (( (*USART).sr & 0x80) == 0)
    ; // wait until it is ok to write
(*USART1).dr = (unsigned short) 'a';
```

Or with the arrow notation:

```
while (( USART->sr & 0x80)==0)
    ; // wait until it is ok to write
USART1->dr = (unsigned short) 'a';
```

USART

Universal synchronous asynchronous receiver transmitter
USART1: 0x40011000
USART2: 0x40004400

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0																																	USART_SR
4																																	USART_DR
8																																	USART_BRR
0xC																																	USART_CR1
0x10																																	USART_CR2
0x14																																	USART_CR3
0x18																																	USART_GTPR

Funktionspekare!

```c
typedef struct TreeNode_t
{
    char * name;
    int active;
    struct TreeNode_t * right;
    struct TreeNode_t * left;
} TreeNode;


void RenameLeafNodes(TreeNode * root)
{
    /*
     * DO A LOT OF STUFF TO FIND THE LEAF NODES
     * AND, FOR EVERY LEAF NODE:
     */
    {
        leaf_node->name = "New Name";
    }
}
```

```c
void DeactivateLeafNodes(TreeNode * root)
{
    /*
     * DO A LOT OF STUFF TO FIND THE LEAF NODES
     * AND, FOR EVERY LEAF NODE:
     */
    {
        leaf_node->active = 0;
    }
}

int main()
{
    TreeNode * root = BuildComplexTree();
    DeactivateLeafNodes(root);
    RenameLeafNodes(root);
}
```

Funktionspekare!

```c
typedef struct TreeNode_t
{
    char * name;
    int active;
    struct TreeNode_t * right;
    struct TreeNode_t * left;
} TreeNode;

typedef void (*LeafNodeFunction)(TreeNode *);

void ProcessLeafNodes(TreeNode * root, LeafNodeFunction lnf)
{
    /*
     * DO A LOT OF STUFF TO FIND THE LEAF NODES
     * AND, FOR EVERY LEAF NODE:
     */
    {
        lnf(leaf_node);
    }
}
```

```c
void RenameLeafNode(TreeNode * leaf_node)
{
    leaf_node->name = "New Name";
}

void DeactivateLeafNode(TreeNode * leaf_node)
{
    leaf_node->active = 0;
}

int main()
{
    TreeNode * root = BuildComplexTree();
    ProcessLeafNodes(root, DeactivateLeafNode);
    ProcessLeafNodes(root, RenameLeafNode);
}
```

Funktionspekare!

```
void f(int x)
{
    printf("%i", x);
}

int main()
{
    void (*fp)(int) = f;
    fp(3);
}
```

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

typedef struct { int health; } Player;

typedef void (*AttackFunction)(Player *);
typedef struct {
    char * name;
    AttackFunction attack;
} Monster;

void EvilAttack(Player * player) {
    player->health -= 10;
    printf("Evil Monster Attacks! Health = %i\n",
           player->health);
}

void FriendlyAttack(Player * player) {
    player->health += 1;
    printf("Friendly monster hugs you! Health = %i\n",
           player->health);
}
```

```c
int main()
{
    ////////////////////////////////////////////////
    // Initialize monsters
    ////////////////////////////////////////////////
    const int NUM_MONSTERS = 100;
    Monster monsters[NUM_MONSTERS];
    for(int i=0; i<NUM_MONSTERS; i++) {
        if(rand()%5 == 0) { // Create evil monster
            monsters[i].name = "Evil Monster";
            monsters[i].attack = EvilAttack;
        }
        else { // Create friendly monster
            monsters[i].name = "Friendly Monster";
            monsters[i].attack = FriendlyAttack;
        }
    }
    ////////////////////////////////////////////////
    // Start fighting!
    ////////////////////////////////////////////////
    Player player = {.health = 1000};
    int current_monster = 0;
    while(player.health > 0) {
        monsters[current_monster].attack(&player);
        current_monster = (current_monster + 1) % NUM_MONSTERS;
    }
    printf("Oh no! You died.\n");
}
```