Dag 3

Pedher Johansson (pedher@cs.umu.se)
Thomas Johansson (thomasj@cs.umu.se)
Mattias Åsander (mattiasa@cs.umu.se)

Pekare

- Lagrar minnesadress
- Minnesutrymmet *allokeras* separat

```
int var;

var = 4;
```

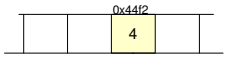
```
int *var;
var = new int;

*var = 4;
```

OBS

4

0x44f2



Allokera minne

- Minne av en viss datatyp reserveras med **`new`**
- Reserveras tills
  - programmet avslutas
  - **`delete`** används

```
int *var;
var = new int;

*var = 4;

delete var;
```


`int[]`

```
int var[2];

var[0] = 3;
var[1] = 20;

cout << var[0] << endl;
```

```
int *var = new int[2];

var[0] = 3;
var[1] = 20;
cout << var[0] << endl;

delete[] var;
```

`struct`

```
timestamp_t t;

t.hours   = 3;
t.minutes = 20;

cout << t.hours << endl;
```

```
timestamp_t *t;
t = new timestamp_t;

t->hours   = 3;
t->minutes = 20;
cout << t->hours << endl;

delete t;
```

`class`

```
Timestamp t(3, 20);

cout << t.getHours() << endl;
```

```
Timestamp *t;
t = new Timestamp(3, 20);

cout << t->getHours() << endl;

delete t;
```


Tilldela pekare

- Vid tilldelning kopieras *värdet* (nu en adress), precis som vid vanlig tilldelning.
  - Flera pekare till samma värde

```
int v1;
int v2;

p1 = 4;
p2 = p1;
```

```
int* p1;
int* p2;

p1 = new int;
p2 = p1;

delete p1;
```


Varför (inte) dynamiskt minne?

- Data kopieras inte – en instans
  - Effektivitet
  - Konsistens
- Kan allokeras efter behov
  - Ibland nödvändigt
- Risk för minnesläckor/pekarfel
  - Svårdebuggat

Destruktor

- Minnesläckor är förödande
- Lokalt ansvar för allokerat minne
  - Klasser som allokerar minne, får ansvar att avallokera
- Destruktor anropas när ett objekt avallokeras
  - Får ansvar att städa upp efter objektet


```
class IntArray
{
private:
    int *array;
public:
    IntArray(int length)
    {
        array = new int[length];
    }
    void set(int index, int value)
    {
        array[index] = value;
    }
    ~IntArray()
    {
        delete[] array;
    }
};

int main()
{
    IntArray *v = new IntArray(10);
    v->set(0, 30);
    delete v;

    return 0;
}
```


Att länka program

Kompilering och länkning

- Vid kompilering skapas maskinkod av en fil
  - Behöver vet hur en funktion ska användas, men inte vad den gör eller var den finns
- Vid länkningen anges var en funktion finns för ett visst anrop
  - I programmet eller i externa bibliotek
- Först kompileras varje fil, sedan länkas programmet.

Multipla kopior

Källkod är (nästan) alltid uppdelad på flera filer, men

1) något måste vara deklarerat innan det används
2) vi kan inte ha kopior

Höna och ägg om vi vill använda något i flera filer.

Prototyper

En fil delas i två

- En ”header”-fil med prototyper och typdefinitioner
  - Kan inkluderas i andra filer
- En fil med funktionerna
  - Länkas til program eller bibliotek

```
typedef struct
{
   int hours;
   int minutes;
} timestamp_t;

timestamp_t*
makeTimestamp(int h, int m)
{
   timestamp_t* t;
   t = new timestamp_t;
   t->hours = h;
   t-> minutes = m;
   return t;
}
```

timestamp.h

```
typedef struct
{
   int hours;
   int minutes;
} timestamp_t;

timestamp_t*
makeTimestamp(int, int);
```

calendar.cpp

```
#include "timestamp.h"

timestamp_t*
makeTimestamp(int h, int m)
{
   timestamp_t* t;
   t = new timestamp_t;
   t->hours = h;
   t-> minutes = m;
   return t;
}
```


Headerfilen

- Inte heller prototyper får förekomma fler gånger (t.ex. i fil som inkluderar fil)
- *Preprocessordirektiv* i headerfilen

Om inte `__TIMESTAMP_H__` är definierad

Klipp in följande

```
#ifndef __TIMESTAMP_H__
#define __TIMESTAMP_H__

typedef struct
{
    int hours;
    int minutes;
} timestamp_t;

timestamp_t*
makeTimestamp(int, int);

#endif
```



```
class Timestamp
{
private:
   int hours;
   int minutes;
public:
   Timestamp(int h, int m)
   {
      hours = h;
      minutes = m;
   }
};

int main()
{
   Timestamp *t;
   t = new Timestamp(3, 20);
   delete t;

   return 0;
}
```

timestamp.h

```
#ifndef __TIMESTAMP_H__
#define __TIMESTAMP_H__
class Timestamp
{
private:
    int hours;
    int minutes;
public:
    Timestamp(int h, int m);
};
#endif
```

timestamp.cpp

```
#include "timestamp.h"

Timestamp::Timestamp(int h, int m)
{
   hours = h;
   minutes = m;
}
```

main.cpp

```
#include "timestamp.h"

int main()
{
    Timestamp *t;
    t = new Timestamp(3, 20);
    delete t;

    return 0;
}
```