



Föreläsning 4 Programmeringsteknik och C DD1316

- hashtabeller
- filer
- sårfall (exceptions)



Dictionary (Hashtabell)

- En hashtabell kan ses som en lista där programmeraren associerar egna nycklar (index) till varje element som läggs i tabellen :

Nycklar:	5	1	12
Element:	"maj"	"jan"	"dec"

2/16



Skapa hashtabell

- Skapa med måsvingar och par av nyckel och värde, enligt nedan:

```
ht = {1:"jan", 12:"dec", 5:"maj"}
```

- Skapa med dict():

```
ht = dict()
ht[1] = "jan"
ht[12] = "dec"
ht[5] = "maj"
```

3/16



Slå upp element

- Element slås upp med nyckelobjekt, men det måste finnas ett objekt lagrat med given nyckel:

```
print (ht[1])
print (ht[5])
print (ht[15])
```

Vad skrivs ut av ovanstående?

4/16



Kontrollerad åtkomst

- Alternativ 1: Kontrollera förekomst innan åtkomst.

```
if 15 in ht:
    x = ht[15]
```

- Alternativ 2: Använd get() med default-värde:

```
x = ht.get(15, "FINNS EJ!")
y = ht.get(15)
```

Vad får x och y för värde om 15 inte finns i tabellen?

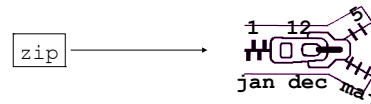
5/16



Skapa hashtabell från listor

- Hashtabell kan skapas från två listor som att dra ihop ett blixtlås:

```
keys = [1, 12, 5]
values = ["jan", "dec", "maj"]
parvis = zip(keys, values)
ht = dict(parvis)
```



6/16



Listor med nycklar och värden

- Metoden `keys()` returnerar en lista med alla nycklar
- Metoden `values()` returnerar en lista med alla värden

```
keyList = list(ht.keys())
valList = list(ht.values())
```

7/16



Ytterligare operationer

- Antalet element i tabellen:
`len(ht)`
- Ta bort element:
`del ht[12]`
- Kopiera hela tabellen:
`htcopy = ht.copy()`

8/16



Öppna fil

En fil kan öppnas för olika ändamål:

- För läsning:
`file = open("filename.txt", "r")`
- För skrivning (skriver över befintlig fil)
`file = open("filename.txt", "w")`
- För tillägg till slutet av filen:
`file = open("filename.txt", "a")`
- Ytterligare alternativ `"r+"`, `"w+"`, `"a+"` fungerar som utan plus fast man kan både skriva och läsa.

9/16



Läsa från öppen fil

- Hela filen:
`contents = file.read()`
- Bestämt antal tecken:
`fiveChars = file.read(5)`
- En rad:
`oneLine = file.readline()`
- Alla rader som en lista:
`allLines = file.readlines()`
- Vid fils slut returneras tomma strängen `('')` vilken har sanningsvärdet falskt och kan användas i villkor

10/16



Skriv till öppen fil och stäng

- Skriv sträng
`file.write("Hejsan hoppsan")`
- Elementen i en lista:
`file.writelines(stringlist)`
- När du läst eller skrivit klart från/till en fil så stänger du den:
`file.close()`

11/16



Loop över raderna i en fil

```
# Följande program läser varje rad
# i filen fil.text i tur och ordning
# och skriver ut den.

file = open('fil.text', 'r')
for line in file:
    # line är här en rad i filen
    print(line)
```



Exception

- `try` och `except` är reserverade ord som används för hantering av exekveringsfel.

```
months = ['jan', 'feb', 'mar', 'maj', 'jun',
          'jul', 'aug', 'sep', 'nov', 'dec']
try:
    i = input("Ange månadsnr: ")
    i = int(i)
    month = months[i - 1]
except:
    print ("Felaktigt månadsnummer")
```

13/16



Syntax

`try:`

Kod som kan orsaka något typ av exekveringsfel

`except:`

Kod som exekveras om och endast om det blir något fel i kodblocket efter `try`

14/16



Så här går det till

Kod i blocket mellan `try:` och `except:` börjar exekvera, men så fort ett fel uppstår i någon rad avbryts exekveringen direkt, resterande rader i blocket exekveras inte och koden som finns i blocket efter `except:` börjar exekvera i stället.

15/16



Olika typer av fel

Typ	Beskrivning
<code>IOError</code>	uppstår när man vill öppna en fil som inte finns
<code>IndexError</code>	uppstår vid användning av ett felaktigt index i en lista
<code>KeyError</code>	uppstår när en nyckel inte finns i hashtabell
<code>TypeError</code>	uppstår när en inbyggd operation eller funktion tillämpas på ett objekt av felaktig typ
<code>ValueError</code>	uppstår när funktion används med ett argument med korrekt typ men fel värde
<code>ZeroDivisionError</code>	uppstår när divisor i en division är 0

16/16



Mer detaljerat

```
months = ['jan', 'feb', 'mar', 'maj', 'jun',
          'jul', 'aug', 'sep', 'nov', 'dec']
try:
    monthNum = input('Ange månadsnummer: ')
    monthNum = int(monthNum)
    parti = plista[monthNum - 1]
except IndexError:
    print ("Ange månadsnummer 1-12")
except ValueError:
    print ("Månadsnummer måste var ett tal")
except:
    print ("Något okänt fel uppstod!")
```

17/16



Sammanfattning

- Hashtabell/Dictionary: snabb uppslagning via nycklar (uppslagning via heltalsindex i lista dock ännu mer effektivt)
- Filer kan öppnas för olika ändamål: läsning, skrivning och tilläggning av data
- `try`, `except` används för att hantera exekveringsfel

18/16