TOD070 VIDEREGÅENDE PROGRAMMERING
Øving 5del_1
Høsten 2010

HØGSKOLEN I BERGEN AVDELING FOR INGENIØRUTDANNING

Øving 5Del-1 Uke 41- 42 innleveringsfrist fredag 22.10 kl.15.00 Sist oppdatert 15.10

Oppgave 1 (strengtabeller og pekere)

a)

Gitt følgende kode under:

```
void strengKopier (const char *fra, char *til){
   while((*til++ = *fra++) != '\0'){
        ;
   }
}
```

Metoden kopierer en tegntstreng (sml biblioteksfunksjonen strcpy side 400). Lag et et lite program som tester ut metoden. Skriv metoden litt mindre "kryptisk og test ut din versjon.

b)

Lag en metode strengLengde som finner lengden av en nullterminert tegnstreng. (sml bibloteksfunksjonen strlen side 400).

```
int strengLengde(const char *tekst){
const char *start = tekst;
   // fyll ut med kode
   }
```

Test ut metoden.

c)

Forklar i detalj hva koden under gjør. Lag en figur av datastrukturene

```
int tabell[antallrekker][antallKolonner];
int *pekerTabell[antallRekker];

for(int i = 0; i < antallRekker; i++){
      pekerTabell[i] = tabell[i];
      for(int j = 0; j < antallKolonner; j++){
            *(pekerTabell[i] + j) = 0;
      }
}
```

d)

Test koden for boblesortering av tall med pekere gitt under før du går videre!

```
void sort(float* p[], int n){
  float* temp;
  for(int i = 1; i < n, i++)
   for(int j = 0; j < n-i; j++)
       if(*p[j] > *p[j]+1]){
           temp = p[j];
           p[j] = p[j+1];
           p[j+1] = temp;
         }
      }
```

Eks på en pekertabell der pekerne peker på strenger (nullterminerte):

```
char* suit[4] = {"Hearts", "Diamonds", "Clubs", "Spades"}
```

Gjør nødvendige endringer i metoden sort over slik at du kan sortere strenger vha pekere:
Tips: metoden må bruke `strcmp`.

```
void sort(char** name, int n)
{
    //Fyll ut
}
```

Test ut metoden

Oppgave 2

a)

i) Forklar kort hva som foregår når vi oppretter et objekt av en klasse som arver fra en annen klasse.
ii) Forklar med egne ord hva som menes med en virtuell metode.
iii) Forklar hva som menes med en ekte virtuell metode.
iv) Forklar hva som menes med tidlig binding og sein binding.
v) Forklar med egne ord hva som menes med polymorfisme.
Kan vi ha polymorfisme uten virtuelle metoder?
vi) Hva menes med en abstrakt klasse?

Vi ser igjen på øving4:

```
class Owner : public Person {
public:
  Owner(const string& n) : Person(n) {};
  void add(Motor_vehicle& vehicle); // "add"
  void erase(Motor_vehicle& vehicle);//"erase"
  void write_info() const;
private:
    Motor_vehicle** carCollection; //alt: vector<Motor_vehicle *> carCollection;
    int capasity;
    int used;
}
```

b)

Hva legger vi opp til i oppgaven når vi skriver:

```
1)Motor_vehicle** carCollection eller
  vector<Motor_vehicle *> carCollection.
```

Hvorfor vil polymorfismen virke (hvis all annen kode er riktig)?

Hvis vi stedet hadde skrevet dette:

```
2) Motor_vehicle* carCollection eller
   vector<Motor_vehicle > carCollection
```

Ville polymorfismen da ha virket? Forklar

c) Under er vist en "demo" av polymorfisme:

```
int main( ){/*

Employee* PTArray[2]; // Ikke dynamiske pekertabell(statisk)
HourlyEmployee joe;  joe.setName("Mighty Joe");
joe.setSsn("123-45-6789");   joe.setRate(20.50);
```

```
    joe.setHours(40);    joe.setNetPay(40*20.50);
    SalariedEmployee boss("Mr. Big Shot", "987-65-4321", 10500.50);

      PTArray[0] = &joe;
      PTArray[1] = &boss;
      PTArray[0]->printCheck();
      PTArray[1]->printCheck();

    return 0;
}
```

`Employee` er en basisklasse for de to andre klassene som har virtuell metode `printCheck()` (`Employee` er en abstrakt klasse, men det behøver den ikke være).

Hvorfor er det bedre å ha pekertabell dynamisk og opprette objektene dynamisk, altså med new-operator?
Tips: Øving 3 oppgave 1. Hva skjer når de variable (pekertabell og objektene) over er definert i en funksjon og vi går ut av funksjonen?

Oppgave 3

Modifiser klassen matrise fra øving 3 slik at unntak kastes og fanges hvis det er feile dimensjoner. Du skal altså ikke bruke exit-metoden.
Lag en egen klasse unntak der du skriver ut en passende melding. Lærebok side 860.
Husk å ta med spesifisering av unntaket i metoden. Lærebok side 864.
se i boken. I testkjøringen skal du vise at unntaket håndteres.

Oppgave 4

a) Hent ned koden for display 16.5 – 16.7 Studer kildekoden. Kompiler og kjør.
Modifiser tilordningsoperator slik at den er mer riktig. Se litt på tilordningsoperator du
lagde for matriseklassen.
Videre, lag en expand()-metode slik at du kan utvide tabellen.
Kompiler og kjør. Test ut programmet.

b) Programming Projects nr 5 side 752

```
template<class T>
class Set
{
public:
      Set();
      void add(T newItem);          // Add new item to the set
      int getSize();                // Return size of set
      T* getArray();                // Return dynamic array of set data
                                    // Caller must delete memory
private:
```

```
        vector<T> data;
};
```

Oppgave 5

Her velger dere motsatt av det dere valgte i øving 4.
Hvis dere i øving 4 gjorde b), så gjør dere c nå og motsatt ellers.

a) Lærebok Programming Project nr 10 lærebok side 933.
   eller
b) Lærebok Programming Project nr 11 lærebok side 933.