

Obligatorisk oppgave nr.2

PROG1003 – Objekt-orientert programmering, våren 2026

Innleveringsfrist: 24.februar 2026 kl.11:00 (*må overholdes*)
i Blackboard på pdf-format

Bruk filen: **OBLIG2.TPL** (les kommentarene nøye, og omdøp den til .CPP)

Det skal lages et program som holder orden på ulike datoer/dager og de ulike aktivitetene som skjer på hver enkelt av dagene. I TPL-filen ser vi at programmet bl.a. inneholder: fire ulike klasser, vectorer både globalt og inni den ene klassen, samt syv funksjoner som kalles fra `main` eller av hverandre.

Funksjonene *skal* utføre følgende:

1. **`void Aktivitet::lesData()`** - Leser inn verdier til begge klassens datamedlemmer.
2. **`void Aktivitet::skrivData()`** - Skriver ut på skjermen de to datamedlemmene (enum-verdier omgjøres til passende tekster).
3. **`void Tidsbegrenset::lesData()`** - Får baseklassen først til å lese inn sine data. Leser deretter inn sine fire datamedlemmer, der *det sikres/sjekkes at klokkeslettene er lovlige* (se funksjon i pkt.7), og at *sluttiden er senere enn starttiden*. Vi forutsetter at ingen aktivitet slutter etter midnatt.
4. **`void Tidsbegrenset::skrivData()`** - Får baseklassen først til å skrive ut sine data. Skriver deretter ut sine egne fire, der ensifrede tall får en innledende null (f.eks: 09:00 – 12:05).
5. **`void Heldags::lesData()`** - Får baseklassen først til å lese inn sine data. Leser deretter inn sitt eget ene datamedlem.
6. **`void Heldags::skrivData()`** - Får baseklassen først til å skrive ut sine data, og deretter sitt eget datamedlem.
7. **`bool Tidsbegrenset::klokkeslettOK(...)`** - Returnerer `true/false` til om parameterne er innenfor intervallene 0-23 og 0-59.

8. **Dag::~~Dag()** - Frigir alt allokert memory av klassen, og tømmer de to vectorene helt.
 9. **bool Dag::harDato(...)** - Returnerer `true/false` til om dagen er den samme datoen som parameterne angir.
 10. **void Dag::nyAktivitet()** - Spør først brukeren om hvilken aktivitet som ønskes opprettet (T(idsbegrenset), H(eldags)). Ut fra valget opprettes aktuelt nytt objekt, dets data leses inn, og den legges inn *bakerst* i aktuell `vector`. **NB:** Det må gjerne foregå flere parallelle aktiviteter (i tid), og flere heldagsaktiviteter på samme dag (vi har altså ingen sjekk på dette).
 11. **void Dag::skrivAktiviteter()** - Skriver ut *alt* innholdet i begge de to vectorene.
 12. **void Dag::skrivDato()** - Skriver *kun* ut dagens dato (dag, måned, år).
 13. **bool dagOK(...)** - Returnerer `true/false` til om parameterne er innenfor intervallene 1-31, 1-12 og 1990-2030. (Du trenger *ikke* å sjekke at datoen *virkelig* er gyldig utover dette. F.eks. at 31/4-1996 egentlig er ulovlig, eller at 29/2-2003 er ulovlig, da dette ikke er et skuddår.)
 14. **Dag* finnDag(...)** - Finner og returnerer evt. en peker til dagen om har en dato lik parametrene (bruk funksjon i pkt.9), ellers returneres `nullptr`.
 15. **void frigiAllokertMemory()** - Sletter *alle* de allokerde dagene, og tømmer vectoren.
 16. **void nyAktivitet()** - Skriver først ut *alle* lagrede datoer (se funksjon i pkt.17). Deretter leses inn en *lovlig* dato (bruk funksjon i pkt.13). Prøver så å finne denne dagen (bruk funksjon i pkt.14). Finnes den *ikke*, opprettes en ny `Dag`, og den legges bakerst i aktuell `vector`. Til slutt leses det inn *en* ny aktivitet på denne (allerede eksisterende eller nyopprettede) dagen.
 17. **void skrivDager(...)** - Kommer en melding om det er tomt for dager. Ellers går det gjennom *alle* de lagrede dagene. Det skrives *alltid* ut dens dato. Om parameteren er `true`, så skrives også ut *alle* dens aktiviteter (vha. funksjonen i pkt.11).
 18. **void skrivEnDag()** - Skriver først ut *alle* lagrede datoer (se funksjon i pkt.17). Deretter leses inn en *lovlig* dato (bruk funksjon i pkt.13). Prøver så å finne denne dagen (bruk funksjon i pkt.14). Finnes den, så skrives *alle* dens aktiviteter, ellers kommer det en melding.
- NB2:** Husk å skrive *ren C++-kode* (og ikke C, med bl.a. `printf`, `scanf`, `gets`, `malloc` og `free`), god og *korrekt* kommentering, innrykk, leselighet og at linjene ikke blir for lange (så teksten wrapper ved en evt. utskrift) - ellers risikerer den dessverre å ikke bli godkjent.

Lykke til!

Malin Foss, William Eide Seiner og FrodeH