



Institutt for datateknikk
og informasjonsvitenskap

Kontinuasjoneksamen i

TDT4102 - Prosedyre- og objektorientert programmering

Fredag 19. august 2011, 09:00

Kontaktperson under eksamen: Hallvard Trætteberg (73593443)

Eksamensoppgaven er utarbeidet av Trond Aalberg

Språkform: Bokmål

Tillatte hjelpemidler: Spesifiserte trykte og håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt (hjelpemiddelkode C).

Tilatte trykte hjelpemidler: Walter Savitch, Absolute C++ eller Lyle Loudon, C++ Pocket Reference

Sensurfrist: Fredag 9 september.

Generell introduksjon

Les gjennom oppgavetekstene og finn ut hva det spørres om. Noen av oppgavene har lengre forklarende tekst, men dette er for å gi mest mulig presis beskrivelse av hva du skal gjøre. Fokuser på det som er det sentrale spørsmålet i hver deloppgave.

All kode skal være C++.

Dersom du mener at opplysninger mangler i en oppgaveformulering, gjør kort rede for de antagelser og forutsetninger som du finner det nødvendig å gjøre. Hver enkelt oppgave er ikke ment å være mer krevende enn det som er beskrevet.

Noen av oppgavene er “oppskriftsbasert” og vi spør etter forskjellige deler av et litt større helhetlig program. Du kan velge selv om du vil løse dette trinnvis ved å ta del for del, eller om du vil lage en samlet implementasjon. Sørg for at det går tydelig frem hvilke spørsmål du har svart på hvor i koden din.

Hele oppgavesettet er arbeidskrevende og det er ikke foreventet at alle skal klare alle oppgaver innen tidsfristen. Tenk strategisk i forhold til ditt nivå og dine ambisjoner!

Oppgavene teller med den andelen som er angitt i prosent. Den prosentvise uttellingen for hver oppgave kan likevel bli justert ved sensur. De enkelte deloppgaver kan også bli tillagt forskjellig vekt.

Oppgave 1: Kodeforståelse og funksjoner (30%)

a) Hva skrives ut av følgende kode?

```
int a = 0, b = 0, c = 0;

a += b + 1;
cout << "1: a = " << a << endl;

b = 5;
a = --b;
cout << "2: a = " << a << endl;

a = 6;
b = 5;
c = a % b;
cout << "3: c = " << c << endl;

a = 5;
b = 6;
c = a / b;
cout << "4: c = " << c << endl;
```

b) Hva skrives ut av følgende kode?

```
int *x = new int(1);
int *y = new int(2);

int z = *x;
cout << "1: z = " << z << endl;

(*y)++;
x = y;
cout << "2: x = " << *x << endl;

*y = 5;
*x = (*y)++;
cout << "3: x = " << *x << endl;
```

- c) Lag en funksjon `int findFirst(int tab[], int val, int size)` som returnerer første posisjon (indeksen) i tabellen `tab` hvor verdien `val` forekommer. Hvilken verdi kan det være hensiktsmessig at funksjonen returnerer hvis verdien ikke finnes i tabellen?
- d) Lag en tilsvarende funksjon `findFirst(...)` for tekststrenger (c-strenger) som kan brukes for å finne indeksen til første forekomst av et tegn (char) i en tekststreng.
- e) Implementer en funksjon `bool isAnagram(char* a, char* b)` som sjekker om to strenger er anagrammer av hverandre. Anagram (av gresk: ana og graphein = omskrive) er et ord, navn eller et fast uttrykk som er blitt satt sammen ved å stokke rundt på bokstavene i et annet ord eller uttrykk. Funksjonen skal ikke ta hensyn til skilletegn (mellomrom, komma, punktum etc.) og skal håndtere store og små bokstaver som like tegn.

Eksempler:

“I am Lord Voldemort” er et anagram av “Tom Marvolo Riddle”

“Balkongdjevelen Kim” er et anagram av “Kjell Magne Bondevik”

- f) I medlemsfunksjonen `dummyfunction` i klassen `Foo` under er nøkkelordet `const` brukt på tre plasser. Forklar hva effekten er for hver enkelt av disse.

```
class Foo{
    private:
        set<string> dummyvar;
    public:
        const string& dummyfunction (const string& p) const;
};
```

Oppgave 2: Funksjoner, unntakstyper, minnehåndtering m.m. (35%)

I denne oppgaven skal du lage et program som slår sammen flere tekstfiler og lagrer disse i ei enkelt fil. Programmet skal først be brukeren skrive inn en serie filnavn det skal leses fra og deretter be brukeren skrive inn et filnavn det skal skrives til.

I `main()` skal det være en variabel `vector<string> infileNames` for å lagre et dynamisk antall filer som programmet skal lese og en variabel `string outfileName` for filnavnet det skal skrives til.

Programmet ditt skal bestå av diverse funksjoner som beskrevet i deloppgavene og eller oppføre seg som spesifisert i deloppgavene. Der parameterlista eller returtypen ikke er spesifisert er det opp til deg å bestemme funksjonsheadere.

- a) Lag en funksjon `void getInfileNames(...)` som brukes for å be brukeren skrive filnavnene som det skal leses fra og lagre disse i `infileNames`. Deretter skal funksjonen be om navnet til fila det skal lagres i. Du kan anta at filnavnene kun inneholder bokstaver og punktumtegnet (f.eks. "testfil.txt"), og eksempelvis lese fra `cin` til en string og avslutte ved at brukeren skriver inn bokstaven 'x'. Her er vi primært ute etter bruk av `cout`, `cin`, kontrollstrukturen du bruker, hvordan du får lagret i `main`-variabelen `infileNames` og `outfile`.
- b) Lag en funksjon `void copy(...)` som leser alle filene i `infileNames` og lagrer filinnholdet fra alle disse filene i `outfileName`.
- c) Utvid programmet med unntakstyper og kasting av unntak. Funksjonen `copy` skal kaste en type unntak hvis en av filene i `infileNames` ikke kan åpnes, og en annen type unntak hvis fila `outfileName` ikke kan åpnes. Begge unntakstypene skal arve fra samme unntakstype og kunne inneholde navnet til fila som forårsaket feilen. Vis hvordan du kan fange opp disse unntakene i `main` og skrive ut forskjellige informasjon avhengig av hvilken type unntak som oppstod.
- d) Lag en ny versjon av funksjonen `copy` hvor du lagrer innholdet i filene i en (eller flere) variabler av dynamisk størrelse (du kan bruke typen `char` slike variabler). Når alle filer er lest (og lagret i minne) skal du lagre alt til fila `outfileName` og frigjøre minne som er brukt før programmet avslutter. Siden filene kan være store, er du i fare for å bruke opp alt tilgjengelig minne. Hvis det ikke er mulig å allokere nok minne vil operatoren `new` kaste et unntak av typen `bad_alloc`. Implementer unntakshåndtering for dette i `main`, men ta også hensyn til at programmet også ved slike unntak skal frigjøre alt allokert minne før det avslutter.

Oppgave 3: Klasser og bruk av STL collection typer (35%)

I appendiks 2 finner du eksempler på bussruter i Trondheim. I denne oppgaven skal du implementere en klasse `BusRoute` som kan brukes for å opprette og manipulere (endre, legge til, slette) slike bussruter.

Viktige elementer i bussruten er:

- Nummeret til bussruten
- Navnet til bussruten
- Tidstabellen over bussavganger. Alle bussruter har en eller flere avganger hver time. I bussruteeksempleen er det bare avganger for timene 06-23, men for enkelthets skyld kan du lage en tidstabell som støtter alle døgnets timer.
- Til enkeltavganger kan det være knyttet noter for mer informasjon om denne avgangen.
- Liste over busstopper med tidligste ankomst (antall minutter etter avgang). Merk at rekkefølgen av disse er viktig.

- Du trenger ikke ta hensyn til annen informasjon enn det som er spesifisert over.

Merk at det kan være nødvendig å lage seg flere klassetyper i tillegg for de enkelte elementene i bussruta og at det kan være hensiktsmessig å lage seg hjelpefunksjoner. I denne oppgaven gjelder det å velge seg hensiktsmessige datatyper for eksempel fra STL for å unngå unødvendig tungvinte løsninger. Les alle deloppgaver før du bestemmer typer.

Lag en klasse som tilfredstiller kravene i deloppgavene under. Det trenger ikke implementere mer enn det som spesifikt er spurt etter i deloppgavene. Der det ikke er spesifisert navn, returtype og parameterliste er det opp til deg å definere funksjonsheaderen.

- a) Medlemsvariabler for nummeret til bussruten og navnet til bussruten og en konstruktør som tar inn navn og nummer og setter disse i en initialiseringsliste.
- b) Get og set-funksjoner for nummeret til bussruten og navnet til bussruten. Det skal kun være mulig å endre nummer og navn ved hjelp av set-funksjonene.
- c) Medlemsvariabel (eller medlemsvariabler) for tidstabellen. Husk at antallet avganger per time kan variere, samt at det kan være en eller flere noter knyttet til hver avgang.
- d) En medlemsfunksjon for å legge til nye avganger `addDeparture(...)`.
- e) En medlemsfunksjon for å legge til en note til en avgang `addDepartureNote(...)`.
- f) En medlemsfunksjon for å slette en avgang `removeDeparture(...)`.
- g) Medlemsvariabel (eller medlemsvariabler) for stoppesteder (husk at dette skal inkludere navnet på stedet og tidspunktet bussen passerer) slik at disse lagres i riktig rekkefølge.
- h) En medlemsfunksjon for å legge til nye busstopper `addStop(...)`.
- i) Gitt at du skal lagre en eller flere bussruter i en variabel av typen `set<BusRoute>`. Hvilke egenskaper har et STL set? Hva må du implementere for klassen for at dette skal kompilere (og fungere).

Appendix 1: Classes and functions that may be of interest

Functions common for all container classes	
begin	Return iterator to beginning (public member type)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
size	Return size (public member function)
empty	Test whether e.g. vector is empty (public member function)
Iterator find (Iterator first, Iterator last, const T& value); Returns an iterator to the first element in the range [first,last) that compares equal to value, or last if not found.	
void sort (Iterator first, Iterator last); Sort elements in range (can be used to sort arrays as well using pointers instead of iterators e.g. sort(tab, tab+10)	
vector	
operator[]	Access element (public member function)
at	Access element (public member function)
front	Access first element (public member function)
back	Access last element (public member function)
push_back	Add element at the end (public member function)
set	
insert	Insert element (public member function)
find	Get iterator to element (public member function)
list	
front	Access first element (public member function)
back	Access last element (public member function)
push_front	Insert element at beginning (public member function)
pop_front	Delete first element (public member function)
push_back	Add element at the end (public member function)
insert	iterator insert (iterator position, const T& x); The list container is extended by inserting new elements before the element at position.
map	

Appendix 1: Classes and functions that may be of interest

operator[]	Access element (public member function) T& operator[] (const key_type& x); Creates a new key-value pair if key does not exist. Example: map<char,string> mymap; mymap['a']="an element"; Alternative access based on map iterator: (*it).first; (*it).second;
insert	Insert element (public member function) Example: mymap.insert (pair<char,int>('z',500));
string	
operator[]	Get character in string (public member function)
at	Get character in string (public member function)
operator+=	Append to string (public member function)
append	Append to string (public member function)
c_str	Get C string equivalent (public member function)
size	Return length of string (public member function)
find	Find content in string (public member function)
rfind	Find last occurrence of content in string (public member function)
find_first_of	Find character in string (public member function)
find_last_of	Find character in string from the end (public member function)
substr	Generate substring (public member function)
Various functions	
istream& getline (istream& is, string& str, char delim); Get line from stream (function)	
istream& getline (istream& is, string& str); Get line from stream (function)	
int get(); Extracts a character from the stream and returns its value (casted to an integer).	
istream& get (char& c); Extracts a character from the stream and stores it in c.	
istream& get (char* s, streamsize n);Extracts a character from the stream and stores it in c.	
int isalnum (int c); Check if character is alphanumeric. A value different from zero is returned (i.e., true) if indeed c is a white-space character. Zero (i.e., false) otherwise.	
int isalpha (int c); Check if character is alphabetic (function)	
int tolower (int c); Convert uppercase letter to lowercase	
int toupper (int c); Convert lowercase letter to uppercase	
int ispunct (int c); Check if character is a punctuation character	
int isspace Check if character is a white-space (function)	

Appendix 2: Bus routes (buss ruter)

46

Pirbadet - Munkegata - Tiller - Sandmoen



Avgangstider fra Pirbadet

Operatør: Nettbuss Trondheim

Time	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
Mandag til fredag	20 35 50	05 20A) 35	05 20 35	05 20 40	00 20 40	00 20 40	00 20 40	00 20 40	00 20 40	05 20 35	05 20 35	05 20 40	00 20 40	00 30	00 30	00 30	00 30	00 30	00 30	00
Lørdag	35	05 35	05 35	05 35	00 20	00 20	00 20	00 20	00 20	00 20	00 20	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00
Søndag					00 30	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00 30	00

A) Holdeplassene fom. KVT tom. Rognbudalen betjenes ikke. Istedet betjenes Tiller v.g.s. og Ivat Skjånes veg. Deretter vanlig trasé fra Ole Ross veg.

Tidligste passering av holdeplassene etter avgang fra Pirbadet 1 minutter. I perioder med mye trafikk vil bussen kunne passere noen minutter senere.

Pirbadet	Follbuva	Brattøra	Polthuset	Trondheim sentralst. hpl. 10	Søndre gate 22	Munkegata M1	Prinsen Kinosen	Studentersamfundet	Emner Tanbarakjelvæes gate	Prof. Brochs gate	Vaknyeggen	Bråsbergvegen	Kroppan bru	Tonsbakryssat	Rossegrenda	John Aaes' veg	City Syd vestre	KVT	City Syd østre	Tonsdalgrenda	Tovmyra	Moltmyra	Moltmyra øst	Ingeborg Olsstads veg	Martin Krognes veg	Kollefåna	Rognbudalen	Ole Ross veg	Torvæket	Edinveien	Djupmyra	Løvassveien	Sandmoen
0	1	1	2	2	3	10	13	14	15	16	17	18	19	22	23	24	24	25	25	26	26	27	27	27	28	29	30	31	32	32	33	34	36

52

Vestlia/Othilienborg - Munkegata



Avgangstider fra Vestlia endeholdeplass

Operatør: Nettbuss Trondheim

Time	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Mandag til fredag	05 35	05 35	05 35	05 35	25	15	05	45	20	20	20	30	20	10	00	40	30	20
Lørdag	30	30	30	30	30	15	15	15	15	15	40	30	20	10	00	40	30	20
Søndag					00 50	40	30	20	10	00	40	30	20	10	00	40	30	20

Tidligste passering av holdeplassene etter avgang fra Vestlia i minutter. I perioder med mye trafikk vil bussen kunne passere noen minutter senere.

Vestlia endeholdeplass	Anne Berggårds veg	Edgar B. Schjødrops veg	Høgagen skole	Nardosenteret	Bjarne Ness veg	Emner Østis veg	K.O. Thornes veg	Othilienborg	K.O. Thornes veg	Emner Østis veg	Tors veg	Nardokryssat	Foløvingen	Dybdals veg	Prestegårdsjordet	Glechaugen syd	Glechaugen nord	Høgskoleinngang	Vatubakken	Studentersamfundet	Prinsen Kinosen	Munkegata M1	
0	0	1	1	2	2	3	3	4	5	6	6	7	8	9	9	9	9	10	11	11	12	13	16

