

Institutt for datateknologi og informatikk

Kontinuasjonseksamensoppgave i PROG1001 – Grunnleggende programmering

Faglig kontakt under eksamen: Frode Haug
Tlf: 950 55 636

Eksamensdato: 14.august 2024
Eksamenstid (fra-til): 09:00-13:00 (4 timer)
Hjelpemiddelkode/Tillatte hjelpemidler: I - Alle trykte og skrevne
(kalkulator er *ikke* tillatt)

Annen informasjon:

Målform/språk: Bokmål
Antall sider (inkl. forside): 8

Informasjon om trykking av eksamensoppgaven

Originalen er:

1-sidig ☒ 2-sidig ☐

sort/hvit ☒ farger ☐

Skal ha flervalgskjema ☐

Kontrollert av:

Dato

Sign

NB: Oppgave 1a, 1b og 2 er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30%)

a) Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):

```
#include <stdio.h>
#include <string.h>

char txt[] =
    "Agios Georgios Agios Nikolaos Agios Gordios Agios Nikitas";

int main() {
    char* t;
    int i = 38, j = i % 24;

    while (txt[i] != '\0') {
        if (txt[i] == 'i') printf("%c ", txt[i-1]);
        ++i;
    }
    printf("\n");

    while (txt[j++] != 'k')
        printf("%c ", txt[j]);
    printf("\n");

    t = &txt[i-j-10];
    while (*t != 'A')
        printf("%c ", *t--);
    printf("\n");

    t = txt;
    while (*t != '\0') {
        if (!strncmp(t, "os ", 3)) printf("%c ", *(t+3));
        t++;
    }
    printf("\n");

    t = txt;
    while (t = strstr(t, "Agios"))
        printf("%c ", *((t++)+8));
    printf("\n");

    return 0;
}
```

b) Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

char txt[][3] = { "us", "er" };

struct Maleri {
    char tittel[40];
    int  bredde, hoyde;
};

void S24Funk1(const struct Maleri m) {
    printf("%s: %ix%i    ", m.tittel, m.bredde, m.hoyde);
}

bool S24Funk2(const struct Maleri m, const int len) {
    return (m.bredde >= len || m.hoyde >= len);
}

bool S24Funk3(const struct Maleri* m) {
    return (strstr(m->tittel, txt[1]) || strstr(m->tittel, txt[0]));
}

struct Maleri S24Funk4(const struct Maleri* m1, const struct Maleri* m2) {
    struct Maleri m;
    strncpy (m.tittel, m1->tittel, 5);      strcat (m.tittel, m2->tittel+4);
    m.bredde = m2->bredde - m1->bredde;      m.hoyde = m2->hoyde - m1->hoyde;
    return m;
}

char S24Funk5(const struct Maleri m1, const struct Maleri* m2) {
    int n1 = strlen(m1.tittel), n2 = strlen(m2->tittel);
    char *t1 = m1.tittel+n1-2, *t2 = m2->tittel+n2-2;

    if (n1 != n2) {
        if (!strcmp(t1, txt[0]) && strcmp(t2, txt[0])) {
            if (m2->bredde > m1.bredde)
                return 'S';
            else return 'M';
        } else return 'A';
    } else return 'I';
}

int main() {
    struct Maleri m1 = { "Bacchus",      85,  98 },
                  m2 = { "Narcissus",    95, 113 },
                  m3 = { "Annunciation", 205, 285 },
                  m4 = { "Musicians",    119,  92 },
                  m5 = { "Lute Player",  205,  98 };

    S24Funk1(m1);   S24Funk1(m2);   S24Funk1(m4);   printf("\n");

    printf("%i %i %i\n", S24Funk2(m3, 260), S24Funk2(m5, 260), S24Funk2(m1, 99));

    printf("%i %i %i\n", S24Funk3(&m3), S24Funk3(&m2), S24Funk3(&m4));

    S24Funk1(S24Funk4(&m4, &m5));   printf("\n");

    printf("%c\n", S24Funk5(m2, &m1));

    return 0;
}
```

Oppgave 2 (70%)

Les *hele* teksten for denne oppgaven (2a-2g) *nøy*e, før du begynner å besvare noe som helst. Studér vedlegget (som også er på utdelte papirer), som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til `#define/const`, structen med datamedlemmer, funksjoner, globale variable, `main()` og `skrivMeny()`. Husk også på de ferdiglagde funksjonene for å lese inn data på `LesData.h`. Bruk *alt* dette *svært* aktivt.

Det skal holdes orden på *ett* idrettslag (fotball, håndball, ishockey, ...) sine resultater i ulike kamper mot (u)like motstandere. Det skal også lagres det unike (innenfor klubben) fornavnet på målscorene.

Datastrukturen

Datastrukturen består (se vedlegget) av arrayen `gKamper`. I denne er indeksene fra 0 til `gAntallKamper-1` i bruk. Vedlegget angir også hvilke datamedlemmer structen inneholder. *Vedlegget inneholder alt du trenger av structer, datamedlemmer og globale variable for å løse denne eksamensoppgaven. Dessuten er **prototyper** for alle funksjoner også ferdig deklarerert/definert.*

Oppgaven

a) Skriv innmaten til `void skrivAlleResultater()` og
`void kampSkrivResultat(const struct Kamp* kamp)`

Den første funksjonen sørger for at det blir gått igjennom alle aktuelle kamper.

Den skriver kampens nummer (fra 1 og oppover), samt får (vha. den andre funksjonen) skrevet ut motstanderlagets navn og resultatet (mål for og imot). Det skrives *en* tekstlinje pr.kamp/resultat. (NB: Målscorenes navn og antall mål skrives ikke her, men i oppgave 2b.)

b) Skriv innmaten til `void skrivEttResultat()` og
`void kampSkrivAlt(const struct Kamp* kamp)`

Den første funksjonen kommer med en egen melding om det ennå ikke er registrert noen kamper.

I motsatt fall leses et *lovlig* kampnummer. Deretter skrives *alle* data ut (vha. den andre funksjonen) om denne kampen. Dvs. det samme skrives ut som i oppgave 2a, *pluss evt.* (alle) målscorenes navn og antall mål (hver på en linje).

c) Skriv innmaten til `void registrerResultat()` og
`void kampRegistrerResultat(struct Kamp* kamp)`

Den første funksjonen kommer med en egen melding om det fullt med kamper. I motsatt fall legges det inn en ny kamp i datastrukturen, og *alle* dens data leses inn vha. den andre funksjonen. Denne funksjonen leser først inn motstanderlagets navn, antall mål selv scoret, antall mål sluppet inn, samt *antall ulike* målscorene på *eget* lag (*ikke* motstanderlaget). Det går så igjennom, og bes om *ett single unikt* (innen laget) fornavn på alle disse egne målscorene, og antall mål vedkommende scoret. Du skal slippe å sjekke at navnet er unikt, men det *må* sørges for at spillerne til sammen ikke har scoret flere mål enn det innledningsvis angitt. Sørg for å håndtere også det at laget har scoret 0 (null) mål.

- d)** **Skriv innmaten til** `void beregnPoengsumOgMaal()` **og**
`void kampHent(const struct Kamp* kamp, int* maalFor, int* maalMot)`
 Den første funksjonen går igjennom alle kampene, beregner og skriver ut *totalt* antall poeng laget har fått (3 poeng for seier, 1 poeng for uavgjort og 0 poeng for tap), og *totalt* antall mål selv scoret og *totalt* antall sluppet inn. Disse to siste verdiene får funksjonen tak i for *hver kamp* ved at den andre funksjonen returnerer dette ved å oppdatere de to siste tilpekte med disse verdiene.
- e)** **Skriv innmaten til** `void beregnPoengsumOgMaal2()`
 Lag en ny versjon av den første funksjonen i oppgave 2d, som ikke bare skriver de tre verdiene bedt om der, men sørger for at det blir skrevet ut en tabellinje som f.eks. ser omtrent slik ut:
- | K | V | U | T | Maal | Poeng |
|---|---|---|---|--------|-------|
| 5 | 3 | 0 | 2 | 15 - 8 | 9 |
- ‘K’ er totalt antall kamper spilt, ‘V U T’ er henholdsvis antall kamper Vunnet, Uavgjort, Tapt. ‘Maal’ og ‘Poeng’ sier seg vel selv, og er det samme som ble beregnet i oppgave 2d.
- f)** **Skriv innmaten til** `void skrivToppscorere()`
 Funksjonen går igjennom *alle* kampene og finner ut hvor mange mål *hver enkelt* spiller *totalt* har scoret. Vi antar laget maksimalt har 30 stk. ulike spillere (selv om max. ulike scorere i *en* kamp altså er 10). Den skriver til slutt ut navnet på spilleren (evt. spillerne) som har scoret aller flest mål. **NB:** Det er her helt greit at *all* din kode er inni funksjonen `skrivToppscorere()`. Dvs. du kan slippe å lage en eller flere funksjoner i tillegg som tar array(er) og/eller `*kamp` som parameter. La altså gjerne bare den ene funksjonen aksessere det den trenger av datamedlemmer i *alle* structene.
- g)** **Skriv innmaten til** `void lesFraFil()` **og**
`void kampLesFraFil(FILE* inn, struct Kamp* kamp)`
 Funksjonene sørger til sammen for at *alle* kampene/resultatene blir lest inn fra filen «KAMPER.DTA». `gAntallKamper` skal ligge aller først på filen.
Filformatet ellers bestemmer du helt selv, men **skal oppgis som en del av besvarelsen.**

Annet (klargjørende):

- Ordene «kamp» og «resultat» brukes synonymt i all teksten ovenfor og i koden (vedlegget).
- Alle målscorene er altså *kun* identifisert via *ett* fornavn, og dette er *unikt innen klubben*.
- Det kan gjerne være flere kamper mot samme lag (altså ingen sjekk på dette i oppgave 2c). Dette skyldes at man selvsagt møtes både til hjemme- og bortekamp - i serien og kanskje i cuper. Det er ingen registrering av om kampen har vært hjemme eller borte.
- Gjør dine egne forutsetninger og presiseringer av oppgaven, dersom du skulle finne dette nødvendig. Gjør i så fall klart rede for disse *i starten* av din besvarelse av aktuell deloppgave.
- **NB:** Det skal *ikke* brukes C++-kode, dvs. slikt som f.eks: string-klassen, kode fra STL, templates eller andre større hjelpebiblioteker. Men, de vanligste inkluder brukt i hele høst er tilgjengelig.

Lykke til med å score mange poeng på denne eksamensoppgaven!
FrodeH

Vedlegg til PROG1001, august 2024: Halvferdig programkode

```
#include <stdio.h>                // printf, FILE
#include <stdlib.h>                // sizeof, malloc
#include <string.h>               // strcpy, strlen, strcmp
#include "LesData.h"              // Verktøykasse for lesing av diverse data

#define MAXSCORERE 10            ///< Max. antall målscorere i EN kamp.
#define MAXKAMPER 40             ///< Max. antall kamper å registrere.
const int STRLEN = 20;           ///< Max. tekstlengde.
const int MAXMAAL = 40;          ///< Max. antall mål scoret/sluppet inn.

/**
 * Kamp (med motstanders navn, antall scorede/innslepne mål og målscorerne).
 */
struct Kamp {                    // En kamps:
    char* navn;                  // - motstandernavn
    int antFor, antMot;           // - antall mål scoret og innslepnet
    int antScorere;               // - antall målscorere
    char* scorere[MAXSCORERE];   // - UNIKT ett-ords fornavn på målscorerne
    int antall[MAXSCORERE];       // - antall mål scoret av nr. 'i'
};

void skrivMeny();                // Ferdiglaget.
void skrivAlleResultater();      // Oppgave 2A
void kampSkrivResultat(const struct Kamp* kamp); // Oppgave 2A
void skrivEttResultat();         // Oppgave 2B
void kampSkrivAlt(const struct Kamp* kamp); // Oppgave 2B
void registrerResultat();        // Oppgave 2C
void kampRegistrerResultat(struct Kamp* kamp); // Oppgave 2C
void beregnPoengsumOgMaal();     // Oppgave 2D
void kampHent(const struct Kamp* kamp, int* maalFor, int* maalMot); //Oppg.2D
void beregnPoengsumOgMaal2();    // Oppgave 2E
void skrivToppscorere();         // Oppgave 2F
void lesFraFil();                // Oppgave 2G
void kampLesFraFil(FILE* inn, struct Kamp* kamp); // Oppgave 2G

int gAntallKamper = 0;           ///< Antall kamper hittil registrert.
struct Kamp* gKamper[MAXKAMPER]; ///< Alle kampene i registeret.

/**
 * Hovedprogrammet:
 */
int main() {
    char kommando;

    lesFraFil();                  // Oppgave 2G

    skrivMeny();
    kommando = lesChar("\nValg");

    while (kommando != 'Q') {
        switch (kommando) {
            case 'A': skrivAlleResultater(); break; // Oppgave 2A
            case 'E': skrivEttResultat(); break; // Oppgave 2B
            case 'R': registrerResultat(); break; // Oppgave 2C
            case 'B': beregnPoengsumOgMaal(); break; // Oppgave 2D
            case 'T': beregnPoengsumOgMaal2(); break; // Oppgave 2E
            case 'S': skrivToppscorere(); break; // Oppgave 2F
            default: skrivMeny(); break;
        }
        kommando = lesChar("\nValg");
    }

    return 0;
}
```

```

/**
 * Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
    printf("\nFOLGENDE KOMMANDOER ER LOVLIG:\n");
    printf("\tA   = skriv noe om Alle kampene/resultatene\n");
    printf("\tE   = skriv alt om En kamp/resultat\n");
    printf("\tR   = Registrer resultatet for en kamp\n");
    printf("\tB   = Beregn og skriv total poengsum og antall maal for/imot\n");
    printf("\tT   = beregn og skriv alt paa en Tabellinje\n");
    printf("\tS   = finn og skriv toppScoreren(e)\n");
    printf("\tQ   = Quit/avslutt\n");
}

/**
 * Oppgave 2A - Skriver ALLE kampene og resultatene fra disse.
 *
 * @see    kampSkrivResultat(...)
 */
void skrivAlleResultater() { /*    LAG INNMATEN    */ }

/**
 * Oppgave 2A - Skriver motstandernavn og antall scorede/innslepne kampmål.
 *
 * @param  kamp - Kampen det skrives ut noen data for
 */
void kampSkrivResultat(const struct Kamp* kamp) { /*    LAG INNMATEN    */ }

/**
 * Oppgave 2B - Skriver ALT om EN gitt kamp.
 *
 * @see    kampSkrivAlt(...)
 */
void skrivEttResultat() { /*    LAG INNMATEN    */ }

/**
 * Oppgave 2B - Skriver ALT om en kamp.
 *
 * @param  kamp - Kampen det skrives ut ALT om
 */
void kampSkrivAlt(const struct Kamp* kamp) { /*    LAG INNMATEN    */ }

/**
 * Oppgave 2C - Legger (om mulig) inn en ny kamp inn i datastrukturen.
 *
 * @see    kampRegistrerResultat(...)
 */
void registrerResultat() { /*    LAG INNMATEN    */ }

/**
 * Oppgave 2C - Leser inn ALT om EN kamp sine resultater.
 *
 * @param  kamp - Kampen det leses inn alle data for
 */
void kampRegistrerResultat(struct Kamp* kamp) { /*    LAG INNMATEN    */ }

```

```

/**
 * Oppgave 2D - Regner og skriver ut TOTALpoengsum og ALLE målene for laget.
 *
 * @see kampHent(...)
 */
void beregnPoengsumOgMaal() { /* LAG INNMATEN */ }

/**
 * Oppgave 2D - Henter antall scorede/innslepne mål i en gitt kamp.
 *
 * @param kamp - Kampen det skal hentes scorede/innslepne mål fra
 * @param maalFor - Antall mål selv scoret ("referanseoverført")
 * @param maalMot - Antall mål innslepnet ("referanseoverført")
 */
void kampHent(const struct Kamp* kamp, int* maalFor, int* maalMot) { /* LAG INNMATEN */ }

/**
 * Oppgave 2E - Lager en "tabellinje" for laget.
 */
void beregnPoengsumOgMaal2() { /* LAG INNMATEN */ }

/**
 * Oppgave 2F - Teller opp og finner navnet til lagets toppscorer(e).
 */
void skrivToppscorere() { /* LAG INNMATEN */ }

/**
 * Oppgave 2G - Leser ALLE kampene/resultatene inn fra fil.
 *
 * @see kampLesFraFil(...)
 */
void lesFraFil() { /* LAG INNMATEN */ }

/**
 * Oppgave 2G - Leser ALT om EN kamp inn fra fil.
 *
 * @param inn - Filen det skal leses inn fra
 * @param kamp - Kampen som får innlest sine data
 */
void kampLesFraFil(FILE* inn, struct Kamp* kamp) { /* LAG INNMATEN */ }

```