TDT4102 – Mai 2018, Utvidet løsningsforslag med noen kommentarer

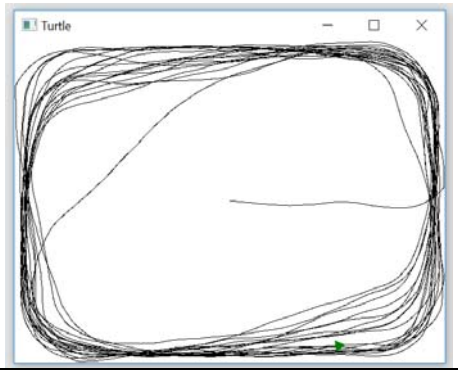
All kode er testet og kjørt, men det er ikke utført veldig grundig testing. (*Grundig testing* av programvare slik det gjøres ved utvikling av kommersielle programvareprodukter har så langt ikke vært vektlagt så mye i dette faget, men er noe vi vil få inn litt mer av i faget i framtiden. Se også kommentar til oppgave 2d).)

Vi presenterer som oftest bare én av mange mulige løsninger som ville ha gitt maksimal uttelling på eksamen, men i noen tilfeller flere alternativer. Vi ber om forståelse for at vi ikke har kapasitet til å lage et LF som viser mange ulike løsnings-muligheter på de ulike delspørsmål. (Husk at alle ikke-trivielle programmeringsoppgaver kan løses på utallige måter.)

I denne versjonen har vi utvidet med noen kommentarer en del steder. Oppgavene ble laget av Lasse og Stipendiat Johannes i tett samarbeid, slik som for også de fire siste eksamener. Vi prøvde gjøre oppgavene litt vanskeligere enn i fjor, og visste at den som vanlig ville være meget arbeidskrevende. Det var kanskje litt flere som klarte å besvare alle deloppgavene enn vi forventet, men vanskelighetsgraden på oppgavene viste seg å stemme med våre forventninger. Det var veldig mange gode besvarelser, som tyder på at mange har lært mye. Karakterene ble omtrent som gj.snitt de siste tre år, men med ca. 5% flere B'er, og 5% færre D'er.

Vi jobber nå med forberedelse av kont.eksamen i august.

God sommer!
17/6-2018, Lasse Natvig



Skjermdump fra oppgave 4.

Generell kommentar om rettingen: I de aller fleste av deloppgavene så har en del detaljer slik som linjeskift, komma osv. svært liten betydning for om studenten klarer å vise at han/hun behersker det vi spør om, og da trekkes det ikke for slike detaljer. Vi trekker heller ikke for mindre syntaks-feil som en kompilator raskt ville ha hjulpet deg med.

Oppgave 1: Kodeforståelse

1a: 33
1b: 21 20 1.05
1c) 2018
1d) 14 Gull 14 Sull 11 Bull
1e) 71
1f) 2 1 3 4 7
1g)

```
s:hihihihi
i:555
```

1h) 42
1i) 32 is nice!

Kommentar: Denne oppgaven ble vanskeligere enn vi antok, da heksadesimal notasjon (som skal være kjent fra IT-GK) har vært mindre synlig i TDT4102 enn vi trodde. Det er litt med i øving 10, og har vært nevnt og vist på forelesning, men det har vært lagt relativt lite vekt på det. Vi har derfor redusert vekten på denne deloppgaven, samt gitt alle studenter minst 50% score på deloppgaven selv om de svarte blankt.

1j)
Bjørgen tok 15
Jansrud tok 2
Klæbo tok 3
Sundby tok 3

Kommentar: Relativt få hadde fått med seg at map er sortert på key, så vi trakk bare litt for det

1k) slurp slurp

Oppgave 2:

2a)

```
bool isLeapYear(int year) {
	return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
}
```

2b)

```
int daysInMonth(int m, int y) {
	int days = months[m - 1];
	if (m == 2 && isLeapYear(y)) {
		days++;
	}
	return days;
}
```

2c)

```
int dayNo(Date day) {
	int numDays = 0;
	for (int m = 1; m < day.m; m++) {
		numDays += daysInMonth(m, day.y);
	}
	return (numDays + day.d);
}
```

Kommentar: Her var det en liten trykkfeil i oppgaveteksten, 2 Februar skulle være 1 Februar. Det hadde ikke betydning ved retting av oppgaven.

2d)

```
Date stringToDate(string str) {
	if (str.size() != 10)
		throw "Invalid date format";
	if ((str[4] != '-') || (str[7] != '-'))
		throw "Invalid date format";
	struct Date d;
	try {
		d.y = stoi(str.substr(0, 4));
		d.m = stoi(str.substr(5, 2));
		d.d = stoi(str.substr(8, 2));
	}
	catch (invalid_argument e) {
		throw "Invalid date format";
	}
	return d;
}
```

Kommentar: (i) Andre løsninger med f.eks. stringstream også fullgode. (ii) Løsninger uten try-catch rundt stoi() godkjennes også fordi stoi kaster et unntak for oss (og vi ikke har sagt hva slags unntak som skal kastes). (iii) Løsningen ovenfor ga full score, men grundigere testing viser at kallet `stringToDate("0aaa-0b-0c")` passerer uten å kaste unntak. I tilfeller som dette praktiserer vi mild retting, som kommer studenten til gode

2e)

```
bool checkDate(Date day) { // (const Date & day) also OK

	return (day.m > 0 && day.m <= 12) &&
		(day.d > 0 && day.d <= daysInMonth(day.m, day.y));
}
```

Kommentar: Det er OK om noen har implementert 2e med const og referanse (&) for bruk i 2f, men det er ikke nødvendig for full score.

2f)

```
string dateToString(Date day) {
	if (!checkDate(day)) {
		return "INVALID DATE";
	}
	stringstream sout;
	sout << setw(4) << setfill('0') << day.y
	<< '-' << setw(2) << day.m << '-' << setw(2) << day.d;
	return sout.str();
}

string dateToStringV2(const Date& d) { // alternativ løsning uten setw/setfill
	if (!checkDate(d)) {
		return "INVALID DATE";
	}
	string str = to_string(d.y);
	int year = d.y;
	while ((year * 10) < 10000) {
		str = "0" + str;
		year *= 10;
	}
	str = str + "-";
	if (d.m < 10) {
		str += "0";
	}
	str += to_string(d.m) + "-";
	if (d.d < 10) {
		str += "0";
	}
	str += to_string(d.d);
	return str;
}
```

2g)

```
struct Event {
	int id;
	string name;
	Date when;
};
ostream & operator<<(ostream & os, const Event & rhs) {
	os << rhs.id << " : " << rhs.name << " @ " << dateToString(rhs.when);
	return os;
}
```

2h)

```
void printEvents(const vector<Event *>& allEvents) {
	for (auto ev : allEvents) {
		cout << *ev << endl;
	}
	cout << allEvents.size() << " events\n";
}
```

Kommentar: Trenger ikke bruke auto eller range-for for full score.

2i)

```
void Calendar::addEvent(int id, string name, int year, int month, int day) {
	if (events.find(id) != events.end()) {
		throw "ID already exists";
	}
	Date d = { year, month, day };
	if (!checkDate(d))
		throw "Invalid Date";
	Event *e = new Event;
	e->id = id;
	e->name = name;
	e->when = d;
	events[id] = e;
}
```

Kommentar: Vi trakk ikke for den minnelekkasje man kan få om new kalles før koden der unntak kan kastes.

2j)

```
Calendar::~Calendar() {
	for (auto e : events)
		delete e.second;
}
```

Kommentar: Trenger ikke bruke auto eller range-for for full score.

2k)

```
vector<Event *> Calendar::getEvents(int year, int month, int day) {
	vector<Event *> evs;
	for (auto it : events) {
		Event *e = it.second;
		if (e->when.y == year && e->when.m == month && e->when.d == day) {
			evs.push_back(e);
		}
	}
	return evs;
}
```

21)

```
vector<Date> Calendar::busyDates(int threshold) {
	map<Date, int> dateFreq;
	for (auto it : events) {
		dateFreq[it.second->when]++;
	}
	vector<Date> dates;
	for (auto& dateEntry : dateFreq) {
		if (dateEntry.second >= threshold) {
			dates.push_back(dateEntry.first);
		}
	}
	return dates;
}

// Nedenfor er en litt mer søkt men også fungerende alternativ løsning med bruk av dateToString
// og stringToDate. Den viser hvordan oppgaven kan løses uten å ha definert operator<

vector<Date> Calendar::busyDates(int threshold) {
	map<string, int> dateFreqMap;
	for (auto it : events) {
		Event *e = it.second;
		dateFreqMap[dateToString(e->when)]++;
	}
	vector<Date> busy;
	for (auto it : dateFreqMap) {
		if (it.second >= threshold) {
			busy.push_back(stringToDate(it.first));
		}
	}
	return busy;
}
```

Oppgave 3:

3a)

```
void Turtle::setPosition(float x, float y) {

	this->x = x;
	this->y = y;
}
sf::Vector2f Turtle::getPosition() {
	return sf::Vector2f(x, y);
}
```

3b)

```
Turtle::Turtle(int width, int height, sf::Color bgColor, sf::Color color)
	: x(0.0), y(0.0), angle(0.0), width(width), height(height), bgColor(bgColor)
{
	if (!canvas.create(width, height)) {
		throw "Failed to create canvas";
	}
	setColor(color);
	setLineWidth(1.0);
	clear();
}
```

Kommentar: Vi ga full score uansett om variabler initialiseres i init-liste eller konstruktørkroppen. Trenger ikke bruke setfunksjoner for å sette medlemsvariable for å få full score.

3c)

```
void Turtle::forward(float distance) {
	float x0 = x;  // Save old position of turtle
	float y0 = y;
	x += distance * cos(deg2rad(angle));
	y += distance * sin(deg2rad(angle));
	drawLine(x0, y0, x, y);
}
```

3d)

```
void Turtle::backward(float distance) {
	forward(-distance);
}
```

Kommentar: Vi ga full score også til løsninger som ikke bruker forward()

```
// Funksjonskropp i alternativ løsning:
	angle += 180.0;
	forward(distance);
	angle -= 180.0;
```

3e)

```
void Turtle::left(float angle) {
	this->angle += angle;
}
void Turtle::right(float angle) {
	this->angle -= angle; // Alternatively: left(-angle);
}
```

3f)

```
void hexagon(Turtle *turtle) {
	turtle->setColor(sf::Color(200, 100, 0));
	for (int i = 0; i < 6; i++) {
		turtle->forward(100);
		turtle->left(60);
	}
}
```

Kommentar: Ikke viktig om man snur til venstre eller høyre for å få full score.

3g)

```
float randRange(float min, float max) {
	return min + (max - min) * (static_cast<float>(rand()) / RAND_MAX);
}

void randomWalk(Turtle *turtle, int steps) {
	for (int i = 0; i < steps; i++) {
		turtle->left(randRange(-90, 90));
		turtle->forward(randRange(0, 10));
	}
}
```

3h)

```
void koch(Turtle *turtle, float length, int level) {
	if (level == 0) {
		turtle->forward(length);
	}
	else {
		length = length / 3;
		level -= 1;
		koch(turtle, length, level);
		turtle->right(60);
		koch(turtle, length, level);
		turtle->left(120);
		koch(turtle, length, level);
		turtle->right(60);
		koch(turtle, length, level);
	}
}
```

Kommentar: Ikke viktig om man tegner med eller mot klokka for å få full score.

3i)

```
void snowflake(Turtle *turtle, float length, int level) {
	for (int i = 0; i < 3; i++) {
		koch(turtle, length, level);
		turtle->left(120);
	}
}
```

Kommentar: Ikke viktig om man går med eller mot klokka, men må samsvare med koch-implementasjonen for å få full score.

Oppgave 4

4a)

```
class Wally : public Turtle {
private:

	float speed; // pixels per second
	float turn;
public:
	void setSpeed(float speed) { this->speed = speed; }
	float getSpeed() { return speed; }
	void setTurn(float turn) { this->turn = turn; }
	float getTurn() { return turn; }
};
```

4b)

```
Wally::Wally(int width, int height, float speed, float turn)
		: Turtle(width, height), speed(speed), turn(turn) {}
```

4c)

```
void Wally::forward(float distance) {
	Turtle::forward(distance);
	if (x > width / 2) {
		x = static_cast<float>(width / 2);
	}
	if (x < -width / 2) {
		x = static_cast<float>(-width / 2);
	}
	if (y > height / 2) {
		y = static_cast<float>(height / 2);
	}
	if (y < -height / 2) {
		y = static_cast<float>(-height / 2);
	}
}
```

Kommentar: Her var vi veldig åpne for ulike løsninger, alt som er innenfor rommet og flytting av roboten er OK. Ikke nødvendig med static_cast for full score.

4d)

```
 void Wally::run() {
	float dist;
	while (true) {
		dist = sense();
		if (dist < 100) {  // approaching a wall, make a turn
			left(turn);
		}
		if (dist > 10) { // still safe to move forward
			forward(speed);
		}
		// No robot is perfect
		left(randRange(-5, 5));
		update(); // // Oppdater tegningen (ikke del av oppgaven)
	}
}
```

Kommentar: Her var det forskjellig tolkning av hva som er «å flytte på seg»: er det kun ved forward eller også ved endring av retning (left). Begge deler er ok for å få full score.

…---ooo0ooo---…