



Sovellusohjelmoinnin jatkokurssi

Oppimispäiväkirja

Janne Lankinen

SISÄLLYS

1	Viikkotehtävät	4
1.1	Teht 1:.....	4
1.1.1	main.cpp.....	4
1.1.2	person.cpp.....	4
1.1.3	person.h	5
1.2	Teht 2:.....	6
1.2.1	main.cpp.....	6
1.2.2	date.cpp.....	6
1.2.3	date.h	8
1.3	Teht 3:.....	9
2	Viikkotehtävät	13
2.1	teht 1	13
2.1.1	main.cpp.....	13
2.1.2	person.cpp:.....	14
2.1.3	person.h	15
2.2	Teht 2.....	16
2.2.1	main.cpp.....	16
2.2.2	person.cpp.....	18
2.2.3	person.h	19
2.3	Teht 3.....	20
2.3.1	main.cpp.....	20
2.3.2	noppa.cpp.....	21
2.3.3	noppa.h	21
2.4	Teht 4.....	22
2.4.1	main.cpp.....	22
2.4.2	noppa.cpp.....	23
2.4.3	noppa.h	25
3	Viikkotehtävät	26
3.1	Teht 1.....	26
3.1.1	main.cpp.....	26
3.1.2	date.cpp.....	26
3.1.3	date.h	29
3.2	teht 2	30
3.2.1	main.cpp.....	30
3.2.2	person.cpp.....	30
3.2.3	person.h	31

3.2.4	address.cpp	32
3.2.5	address.h	33
3.3	Teht 3	33
3.3.1	main.cpp	33
3.3.2	date.cpp	34
3.3.3	date.h	35
3.3.4	calendarentry.cpp	36
3.3.5	calendarentry.h	37
4	Viikkotehtävät	39
4.1	teht 1	39
4.1.1	main.cpp	39
4.1.2	date.cpp	39
4.1.3	date.h	40
4.1.4	calendarentry.cpp	41
4.1.5	calendarentry.h	43
4.2	teht 2	44
4.2.1	main.cpp	44
4.2.2	date.cpp	45
4.2.3	date.h	46
4.2.4	calendarentry.cpp	47
4.2.5	calendarentry.h	49
4.3	teht3	49
4.3.1	main.cpp	49
4.3.2	henkilo.cpp	50
4.3.3	henkilo.h	51
4.4	teht4	51
4.4.1	main.cpp	51
4.4.2	henkilo.cpp	52
4.4.3	henkilo.h	52
5	Viikkotehtävät	54
6	Viikkotehtävät	55

1 Viikkotehtävät

//!!!HUOM. viikolla 3 korjattu huomattuja bugeja aiempaan koodiin

1.1 Teht 1:

1.1.1 main.cpp

```
#include <iostream>
#include <string>
#include "person.h"
using namespace std;

int main(){

    {
        Person Kalle;
        Kalle.setName("Kalle");
        Kalle.setAge(20);

        Person Ville;
        Ville.setName("Ville");
        Ville.setAge(23);

        Kalle.salute();
        Ville.salute();

        int x1 = Kalle.getAge();
        int x2 = Ville.getAge();

        cout << "Kalle is" << x1 << " years old." << endl;
        cout << "Ville is" << x2 << " years old." << endl;
    }

    return 0;
}
```

1.1.2 person.cpp

```
#include "person.h"
#include <iostream>

Person::Person() : name("Unnamed"), age(0) {}

Person::Person(std::string n, int a) : name(n), age(a) {}

Person::~Person() {
    std::cout << "Destructor called for " << name << std::endl;
}

void Person::salute() {
```

```

        std::cout << "Hello, my name is " << name << " and I am " << age << "
years old." << std::endl;
    }

    void Person::setAge(int newAge) {
        if (newAge >= 0) age = newAge;
    }

    int Person::getAge() {
        return age;
    }

    void Person::setName(std::string newName) {
        name = newName;
    }

    std::string Person::getName() {
        return name;
    }

```

1.1.3 person.h

```

#ifndef PERSON_H
#define PERSON_H

#include <string>

class Person {
private:
    std::string name;
    int age;

public:
    Person();
    Person(std::string n, int a);
    ~Person();

    void salute();
    void setAge(int newAge);
    int getAge();
    void setName(std::string newName);
    std::string getName();
};

#endif

```

1.2 Teht 2:

1.2.1 main.cpp

```
#include <iostream>
#include "date.h"

using namespace std;

int main() {
    Date date1;
    date1.askDate();
    date1.printDate();

    date1.addOneDay();
    date1.printDate();

    return 0;
}
```

1.2.2 date.cpp

```
#include "date.h"
#include <iostream>

using namespace std;

void Date::setDate(int newDate) {
    date = newDate;
}

void Date::setMonth(int newMonth) {
    month = newMonth;
}
```

```
void Date::setYear(int newYear) {
    year = newYear;
}

int Date::getDate() {
    return date;
}

int Date::getMonth() {
    return month;
}

int Date::getYear() {
    return year;
}

void Date::printDate() {
    cout << date << "/" << month << "/" << year << endl;
}

void Date::printDate(string format) {}

void Date::askDate() {
    cout << "Enter day: ";
    cin >> date;
    cout << "Enter month: ";
    cin >> month;
    cout << "Enter year: ";
    cin >> year;
}

void Date::addOneDay() {
    date++;
    if (date > 30) {
        date = 1;
    }
}
```

```

        month++;
        if (month > 12) {
            month = 1;
            year++;
        }
    }
}

```

1.2.3 date.h

```

#ifndef DATE_H
#define DATE_H

#include <string>

class Date {
private:
    int date;
    int month;
    int year;

public:
    void setDate(int newDate);
    void setMonth(int newMonth);
    void setYear(int newYear);

    int getDate();
    int getMonth();
    int getYear();

    void printDate();
    void printDate(std::string format);
    void askDate();
    void addOneDay();
};

```



```
#endif
```

1.3 Teht 3:

main.cpp:

```
#include <iostream>
#include <string>
#include "date.h"
#include "date.cpp"
using namespace std;

int main() {
    Date date1;
    date1.askDate();
    date1.printDate();

    date1.addOneDay();
    date1.printDate();

    return 0;
}
```

date.cpp:

```
#include "date.h"
#include <iostream>

using namespace std;

class Date {
public:
    void setDate(int newDate);
```

```
int getDate();
void setMonth(int newMonth);
int getMonth();
void setYear(int newYear);
int getYear();
void printDate();
void printDate(string format);
void askDate();
void addOneDay();
private:
    int date;
    int month;
    int year;
};

void Date::setDate(int newDate) {
    date = newDate;
}

int Date::getDate() {
    return date;
}

void Date::setMonth(int newMonth) {
    month = newMonth;
}

int Date::getMonth() {
    return month;
}

void Date::setYear(int newYear) {
    year = newYear;
}
```

```

int Date::getYear() {
    return year;
}

void Date::printDate() {
    cout << date << "/" << month << "/" << year << endl;
}

void Date::printDate(string format) {
    // Implement custom format printing if needed
}

void Date::askDate() {
    cout << "Enter day: ";
    cin >> date;
    cout << "Enter month: ";
    cin >> month;
    cout << "Enter year: ";
    cin >> year;
}

void Date::addOneDay() {
    date++;
    if (date > 30) { // Simplified month length handling
        date = 1;
        month++;
        if (month > 12) {
            month = 1;
            year++;
        }
    }
}

```

date.h:

```
#include <string>

using namespace std;

class Date {
    private:
        int date;
        int month;
        int year;

    public:
        void setDate(int newDate);
        void setMonth(int newMonth);
        void setYear(int newYear);
        int getDate();
        int getmMonth();
        int getYear();
        void printDate(string format);
        void printDate();
        void askDate();
        void addOneDay();
};
```

2 Viikkotehtävät

2.1 teht 1

2.1.1 main.cpp

```
#include <iostream>
#include <string>
#include "person.h"
using namespace std;

int main(){

    setlocale(LC_ALL, "fi_FI");

    {
        Person Kalle;
        Kalle.setName("Kalle");
        Kalle.setAge(20);

        Person Ville;
        Ville.setName("Ville");
        Ville.setAge(23);

        Kalle.salute();
        Ville.salute();

        cout << "for lohkon sisällä luotu olio" << endl;
        for (int i = 0; i < 2; i++){
            Person tempPerson("Jalmari", i + 20);
            tempPerson.printPersonDetails();
        }

        cout << "Aliohjelman sisällä luotu olio" << endl;
```

```

int x1 = Kalle.getAge();
int x2 = Ville.getAge();

cout << "Kalle is" << x1 << " years old." << endl;
cout << "Ville is" << x2 << " years old." << endl;

Person Jalmari("Jalmari", 20);
Jalmari.printPersonDetails();
}

return 0;
}

```

2.1.2 person.cpp:

```

#include "person.h"

Person::Person() {
    name = "Unknown";
    age = 0;
}

Person::Person(std::string name, int age) {
    this->name = name;
    this->age = age;
}

Person::~~Person() {
    std::cout << "Person " << name << " is being destroyed." << std::endl;
}

void Person::setName(std::string name) {

```

```

        this->name = name;
    }

    void Person::setAge(int age) {
        this->age = age;
    }

    std::string Person::getName() {
        return name;
    }

    int Person::getAge() {
        return age;
    }

    void Person::salute() {
        std::cout << "Hello, my name is " << name << "!" << std::endl;
    }

    void Person::printPersonDetails() {
        std::cout << "Name: " << name << ", Age: " << age << std::endl;
    }

```

2.1.3 person.h

```

#ifndef PERSON_H
#define PERSON_H
#include <string>
#include <iostream>
using namespace std;

class Person {
    private:

```

```

        string name;
        int age;
    public:
        Person();
        Person(string n, int a);
        ~Person();

        void printPersonDetails();
        void salute();
        void setAge(int newAge);
        int getAge();
        void setName(string newName);
        string getName();
};

#endif

```

2.2 Teht 2

2.2.1 main.cpp

```

#include <iostream>
#include <string>
#include "person.h"
using namespace std;

void createPerson() {
    Person tempPerson("Temporary", 30);
    tempPerson.printPersonDetails();
}

int main(){

    setlocale(LC_ALL,"fi_FI");

```



```
cout << "Creating person at the beginning of main" << endl;
Person Kalle;
Kalle.setName("Kalle");
Kalle.setAge(20);

cout << "Creating person inside if block" << endl;
if (true) {
    Person Ville;
    Ville.setName("Ville");
    Ville.setAge(23);
    Ville.salute();
}

cout << "Creating person inside for loop" << endl;
for (int i = 0; i < 2; i++){
    Person tempPerson("Jalmari", i + 20);
    tempPerson.printPersonDetails();
}

cout << "Creating person inside a function" << endl;
createPerson();

cout << "Creating dynamic person" << endl;
Person* pekka = new Person("Pekka", 20);
pekka->printPersonDetails();
delete pekka;

cout << "End of main" << endl;

return 0;
}
```

2.2.2 person.cpp

```
#include "person.h"
```

```
Person::Person() {  
    cout << "Person class default constructor" << endl;  
    name = "";  
    age = 0;  
}
```

```
Person::Person(string name, int age) {  
    cout << "Person class parameterized constructor" << endl;  
    this->name = name;  
    this->age = age;  
}
```

```
Person::~Person() {  
    cout << "Person class destructor for " << name << endl;  
}
```

```
void Person::setName(string name) {  
    this->name = name;  
}
```

```
void Person::setAge(int age) {  
    this->age = age;  
}
```

```
string Person::getName() {  
    return name;  
}
```

```
int Person::getAge() {  
    return age;  
}
```

```

void Person::salute() {
    cout << "Hello, my name is " << name << " and I am " << age << " years old."
    << endl;
}

```

```

void Person::printPersonDetails() {
    cout << "Name: " << name << ", Age: " << age << endl;
}

```

2.2.3 person.h

```

#ifndef PERSON_H
#define PERSON_H

#include <string>
#include <iostream>
using namespace std;

class Person {
public:
    Person();
    Person(string name, int age);
    ~Person();
    void setName(string name);
    void setAge(int age);
    string getName();
    int getAge();
    void salute();
    void printPersonDetails();

private:
    string name;
    int age;
};

```

```
#endif
```

2.3 Teht 3

2.3.1 main.cpp

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
#include "noppa.h"

using namespace std;

int main () {

    setlocale(LC_ALL, "fi_FI");

    srand(time(0));

    Cube Noppa;

    int latestThrow = Noppa.throwCube();
    Noppa.showLatestThrow();

    return 0;
}
```

2.3.2 noppa.cpp

```
#include "noppa.h"

Cube::Cube() {
    latestThrow = 0;
}

Cube::~Cube() {
    std::cout << "cube object destroyed" << std::endl;
}

int Cube::throwCube() {
    latestThrow = rand() % 6 + 1;
    return latestThrow;
}

void Cube::showLatestThrow() {
    std::cout << "Latest throw: " << latestThrow << std::endl;
}
```

2.3.3 noppa.h

```
#ifndef NOPPA_h
#define NOPPA_h
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
using namespace std;

class Cube {
private:
    int latestThrow;
public:
```

```

    Cube();
    ~Cube();

    int throwCube();
    void showLatestThrow();
};

#endif

```

2.4 Teht 4

2.4.1 main.cpp

```

#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
#include "noppa.h"

using namespace std;

int main () {

    setlocale(LC_ALL, "fi_FI");

    int gameChoice;
    cout << "Choose a game: 1. Monopoly 2. Yatzy" << endl;
    cin >> gameChoice;

    if (gameChoice == 1) {
        Cube monopoly(2);
        monopoly.throwCube();
        monopoly.showLatestThrow();
    }
    else if (gameChoice == 2) {

```

```

    Cube yatzy(5);
    yatzy.throwCube();
    yatzy.showLatestThrow();
    std::cout << "results";
}

```

```

srand(time(0));

```

```

Cube Noppa;

```

```

Noppa.throwCube();
Noppa.showLatestThrow();

```

```

    return 0;
}

```

2.4.2 noppa.cpp

```

#include "noppa.h"
#include <iostream>
#include <string>
#include <ctime>

```

```

Cube::Cube() : numDice(1) {
    std::cout << "cube class default constructuor" << std::endl;
    srand(time(0));
}

```

```

Cube::Cube(int numDice) : numDice(numDice) {
    std::cout << "cube parameterized constructor" << std::endl;
    srand(time(0));
}

```

```

Cube::~Cube () {}

```

```

void Cube::setNumDice(int numDice) {
    if (numDice >= 1 && numDice <= 5) {
        this->numDice = numDice;
    }
    else {
        std::cout << "Invalid number of dice. Number must be between 1-5" <<
std::endl;
    }
}

int Cube::getNumDice() {
    return numDice;
}

void Cube::throwCube() {
    latestThrows.clear();
    for (int i = 0; i < numDice; ++i) {
        latestThrows.push_back(rand() % 6 + 1);
    }
}

void Cube::showLatestThrow() {
    int sum = 0;
    for (int i = 0; i < latestThrows.size(); ++i) {
        cout << "Dice " << i + 1 << ": " << latestThrows[i] << endl;
        sum += latestThrows[i];
    }
    cout << "Total: " << sum << ". Thrown with " << numDice << " dice." << endl;
}

```


2.4.3 noppa.h

```
#ifndef NOPPA_H
#define NOPPA_H

#include <iostream>
#include <vector>
using namespace std;

class Cube {
private:
    int numDice;
    vector<int> latestThrows;

public:
    Cube(); // Default const
    Cube(int numDice);
    ~Cube();

    void setNumDice(int numDice);
    int getNumDice();
    void throwCube();
    void showLatestThrow();
};

#endif
```

3 Viikkotehtävät

3.1 Teht 1

3.1.1 main.cpp

```
#include "date.h"

int main() {
    Date date1;

    date1.askDate();

    date1.printDate();

    date1.printDate("DD-MM-YYYY");
    date1.printDate("YYYY/MM/DD");

    date1.addOneDay();
    date1.printDate();

    return 0;
}
```

3.1.2 date.cpp

```
#include "date.h"
#include <iostream>
#include <iomanip>

using namespace std;

// Default constructor
Date::Date() {
    day = 1;
    month = 1;
    year = 2000;
}

// Setters
```

```

void Date::setDay(int newDay) {
    day = newDay;
}

void Date::setMonth(int newMonth) {
    month = newMonth;
}

void Date::setYear(int newYear) {
    year = newYear;
}

// Getters
int Date::getDay() {
    return day;
}

int Date::getMonth() {
    return month;
}

int Date::getYear() {
    return year;
}

// Print functions
void Date::printDate() {
    cout << setw(2) << setfill('0') << day << "/"
         << setw(2) << setfill('0') << month << "/"
         << year << endl;
}

void Date::printDate(string format) {
    if (format == "DD-MM-YYYY") {
        cout << setw(2) << setfill('0') << day << "-"
             << setw(2) << setfill('0') << month << "-"
             << year << endl;
    }
    else if (format == "YYYY/MM/DD") {
        cout << year << "/"
             << setw(2) << setfill('0') << month << "/"
             << setw(2) << setfill('0') << day << endl;
    }
    else {
        printDate();
    }
}

void Date::askDate() {
    bool valid = false;
    while (!valid) {

```

```

        cout << "Enter day: ";
        cin >> day;
        cout << "Enter month: ";
        cin >> month;
        cout << "Enter year: ";
        cin >> year;

        if (month >= 1 && month <= 12 && day >= 1 && day <= daysInMonth(month, year)) {
            valid = true; // Valid date
        } else {
            cout << "Invalid date! Please try again." << endl;
        }
    }
}

void Date::addOneDay() {
    day++;

    if (day > daysInMonth(month, year)) {
        day = 1;
        month++;
        if (month > 12) {
            month = 1;
            year++;
        }
    }
}

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

int Date::daysInMonth(int month, int year) {
    const int daysInEachMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30,
    31, 30, 31};

    // Leap year check for February
    if (month == 2 && isLeapYear(year)) {
        return 29;
    }

    return daysInEachMonth[month - 1];
}

```

3.1.3 date.h

```
#ifndef DATE_H
#define DATE_H

#include <iostream>
#include <string>
using namespace std;

class Date {
private:
    int day;
    int month;
    int year;

public:
    Date();

    // Setters
    void setDay(int newDay);
    void setMonth(int newMonth);
    void setYear(int newYear);

    // Getters
    int getDay();
    int getMonth();
    int getYear();

    // Print functions
    void printDate();
    void printDate(string format);

    // Other
    void askDate();
    void addOneDay();

private:
    bool isLeapYear(int year);
    int daysInMonth(int month, int year);
};

#endif
```

3.2 teht 2

3.2.1 main.cpp

```
#include <iostream>
#include "person.h"
#include "address.h"

using namespace std;

int main() {

    Address address1("Example Street 12", "00100", "Helsinki");

    Person person1("John Doe", 30, address1);

    Person person2("Anna Smith", 25);

    cout << "First person's details:" << endl;
    person1.printDetails();
    cout << endl;

    Address address2("Kotikatu 5", "00200", "Espoo");
    person2.setAddress(address2);

    cout << "Second person's details:" << endl;
    person2.printDetails();

    return 0;
}
```

3.2.2 person.cpp

```
#include "person.h"

Person::Person() : name(""), age(0) {}

Person::Person(string name, int age) {
    this->name = name;
    this->age = age;
}

Person::Person(string name, int age, Address address) {
    this->name = name;
    this->age = age;
    this->address = address;
}

void Person::setName(string name) {
```

```

        this->name = name;
    }

    void Person::setAge(int age) {
        this->age = age;
    }

    void Person::setAddress(Address address) {
        this->address = address;
    }

    string Person::getName() {
        return name;
    }

    int Person::getAge() {
        return age;
    }

    Address Person::getAddress() {
        return address;
    }

    void Person::printDetails() {
        cout << "Name: " << name << ", Age: " << age << " years" << endl;
        cout << "Address: ";
        address.printDetails();
    }

```

3.2.3 person.h

```

#ifndef PERSON_H
#define PERSON_H

#include <iostream>
#include <string>
#include "address.h"

using namespace std;

class Person {
private:
    string name;
    int age;
    Address address;

public:
    Person();
    Person(string name, int age);
    Person(string name, int age, Address address);

```

```

    void setName(string name);
    void setAge(int age);
    void setAddress(Address address);

    string getName();
    int getAge();
    Address getAddress();

    void printDetails();
};

#endif

```

3.2.4 address.cpp

```

#include "address.h"

Address::Address() : street(""), postalCode(""), city("") {}

Address::Address(string street, string postalCode, string city) {
    this->street = street;
    this->postalCode = postalCode;
    this->city = city;
}

void Address::setStreet(string street) {
    this->street = street;
}

void Address::setPostalCode(string postalCode) {
    this->postalCode = postalCode;
}

void Address::setCity(string city) {
    this->city = city;
}

string Address::getStreet() {
    return street;
}

string Address::getPostalCode() {
    return postalCode;
}

string Address::getCity() {
    return city;
}

```



```
void Address::printDetails() {
    cout << "Address: " << street << ", " << postalCode << " " << city << endl;
}
```

3.2.5 address.h

```
#ifndef ADDRESS_H
#define ADDRESS_H

#include <iostream>
#include <string>
using namespace std;

class Address {
private:
    string street;
    string postalCode;
    string city;

public:
    Address(); // Default constructor
    Address(string street, string postalCode, string city);

    void setStreet(string street);
    void setPostalCode(string postalCode);
    void setCity(string city);

    string getStreet();
    string getPostalCode();
    string getCity();

    void printDetails();
};

#endif
```

3.3 Teht 3

3.3.1 main.cpp

```
#include <iostream>
#include "CalendarEntry.h"
#include "Date.h"

using namespace std;

int main() {
```

```

    cout << "Default constructor test:" << endl;
    CalendarEntry entry1;
    entry1.printEntry();
    cout << endl;

    cout << "Parameterized constructor test:" << endl;
    Date d(15, 4, 2024);
    CalendarEntry entry2(d, "Meeting at 10 AM", true);
    entry2.printEntry();
    cout << endl;

    cout << "User input calendar entry:" << endl;
    CalendarEntry entry3;
    entry3.askDetails();
    cout << endl;
    entry3.printEntry();

    return 0;
}

```

3.3.2 date.cpp

```

#include "Date.h"

// Default constructor
Date::Date() {
    day = 1;
    month = 1;
    year = 2000;
}

// Parameterized constructor
Date::Date(int d, int m, int y) {
    day = d;
    month = m;
    year = y;
}

// Getter methods
int Date::getDay() const {
    return day;
}

int Date::getMonth() const {
    return month;
}

int Date::getYear() const {
    return year;
}

```

```

// Setter methods
void Date::setDay(int d) {
    day = d;
}

void Date::setMonth(int m) {
    month = m;
}

void Date::setYear(int y) {
    year = y;
}

void Date::printDate() const {
    cout << day << "/" << month << "/" << year << endl;
}

void Date::askDate() {
    cout << "Enter day: ";
    cin >> day;
    cout << "Enter month: ";
    cin >> month;
    cout << "Enter year: ";
    cin >> year;
}

```

3.3.3 date.h

```

#ifndef DATE_H
#define DATE_H

#include <iostream>
#include <string>

using namespace std;

class Date {
private:
    int day;
    int month;
    int year;

public:
    // Constructors
    Date(); // Default constructor
    Date(int d, int m, int y); // Parameterized constructor

```

```

    // Getter methods
    int getDay() const;
    int getMonth() const;
    int getYear() const;

    void setDay(int d);
    void setMonth(int m);
    void setYear(int y);

    void printDate() const;

    void askDate();
};

#endif

```

3.3.4 calendareentry.cpp

```

#include "CalendarEntry.h"

// Default constructor
CalendarEntry::CalendarEntry() {
    date = Date();
    subject = "Not defined";
    reminder = false;
}

// Parameterized constructor
CalendarEntry::CalendarEntry(Date d, string s, bool r) {
    date = d;
    subject = s;
    reminder = r;
}

// Destructor
CalendarEntry::~CalendarEntry() {
    cout << "Calendar entry deleted." << endl;
}

// Getters
Date CalendarEntry::getDate() const {
    return date;
}

string CalendarEntry::getSubject() const {
    return subject;
}

bool CalendarEntry::getReminder() const {

```

```

        return reminder;
    }

    // Setters
    void CalendarEntry::setDate(Date d) {
        date = d;
    }

    void CalendarEntry::setSubject(string s) {
        subject = s;
    }

    void CalendarEntry::setReminder(bool r) {
        reminder = r;
    }

    // Print the calendar entry
    void CalendarEntry::printEntry() {
        cout << "Date: ";
        date.printDate();
        cout << "Subject: " << subject << endl;
        cout << "Reminder: " << (reminder ? "On" : "Off") << endl;
    }

    // Ask the user for calendar entry details
    void CalendarEntry::askDetails() {
        cout << "Enter the calendar entry date: " << endl;
        date.askDate();
        cout << "Enter the subject of the entry: ";
        cin.ignore(); // Clears the input
        getline(cin, subject);
        char reminderChoice;
        cout << "Is the reminder on? (y/n): ";
        cin >> reminderChoice;
        reminder = (reminderChoice == 'y' || reminderChoice == 'Y');
    }
}

```

3.3.5 calendareentry.h

```

#ifndef CALENDAREENTRY_H
#define CALENDAREENTRY_H

#include <iostream>
#include <string>
#include "Date.h"

using namespace std;

class CalendarEntry {
private:

```

```
    Date date;
    string subject;
    bool reminder;

public:
    CalendarEntry();

    CalendarEntry(Date d, string s, bool r);

    ~CalendarEntry();

    Date getDate() const;
    string getSubject() const;
    bool getReminder() const;

    void setDate(Date d);
    void setSubject(string s);
    void setReminder(bool r);

    void printEntry();
    void askDetails();
};

#endif
```

4 Viikkotehtävät

4.1 teht 1

4.1.1 main.cpp

```
#include <iostream>
#include "CalendarEntry.h"
#include "Date.h"

using namespace std;

int main() {
    // original CalendarEntry
    Date date(12, 4, 2024);
    CalendarEntry entry1(date, "Meeting at 10 AM", true);

    cout << "Original Entry:" << endl;
    entry1.printEntry();

    // create a copy using the copy constructor
    CalendarEntry entry2 = entry1; //copy constructor

    cout << "\nCopied Entry:" << endl;
    entry2.printEntry();

    return 0;
}
```

4.1.2 date.cpp

```
#include "Date.h"

// Default constructor
Date::Date() {
    day = 1;
    month = 1;
    year = 2000;
}

// Parameterized constructor
Date::Date(int d, int m, int y) {
    day = d;
    month = m;
    year = y;
}

// Getter methods
int Date::getDay() const {
    return day;
}
```

```

int Date::getMonth() const {
    return month;
}

int Date::getYear() const {
    return year;
}

// Setter methods
void Date::setDay(int d) {
    day = d;
}

void Date::setMonth(int m) {
    month = m;
}

void Date::setYear(int y) {
    year = y;
}

void Date::printDate() const {
    cout << day << "/" << month << "/" << year << endl;
}

void Date::askDate() {
    cout << "Enter day: ";
    cin >> day;
    cout << "Enter month: ";
    cin >> month;
    cout << "Enter year: ";
    cin >> year;
}

```

4.1.3 date.h

```

#include "Date.h"

// Default constructor
Date::Date() {
    day = 1;
    month = 1;
    year = 2000;
}

// Parameterized constructor
Date::Date(int d, int m, int y) {

```



```

        day = d;
        month = m;
        year = y;
    }

    // Getter methods
    int Date::getDay() const {
        return day;
    }

    int Date::getMonth() const {
        return month;
    }

    int Date::getYear() const {
        return year;
    }

    // Setter methods
    void Date::setDay(int d) {
        day = d;
    }

    void Date::setMonth(int m) {
        month = m;
    }

    void Date::setYear(int y) {
        year = y;
    }

    void Date::printDate() const {
        cout << day << "/" << month << "/" << year << endl;
    }

    void Date::askDate() {
        cout << "Enter day: ";
        cin >> day;
        cout << "Enter month: ";
        cin >> month;
        cout << "Enter year: ";
        cin >> year;
    }
}

```

4.1.4 calendarentry.cpp

```
#include "CalendarEntry.h"
```

```

// Default constructor
CalendarEntry::CalendarEntry() {
    date = Date(); // Default to 1/1/2000
    subject = "Not defined";
    reminder = false;
}

// Parameterized constructor
CalendarEntry::CalendarEntry(const Date& d, const string& s, bool r) {
    date = d;
    subject = s;
    reminder = r;
}

// Copy constructor (Kopiorakentaja)
CalendarEntry::CalendarEntry(const CalendarEntry& other) {
    // Kopioidaan vain subject ja reminder
    subject = other.subject;
    reminder = other.reminder;

    // Kysytään uusi päivämäärä käyttäjältä
    cout << "Enter new date for the copied entry (day month year): ";
    int day, month, year;
    cin >> day >> month >> year;
    date = Date(day, month, year); // Luo uusi päivämäärä
}

// Destructor
CalendarEntry::~CalendarEntry() {
    cout << "Calendar entry deleted." << endl;
}

// Getter methods
Date CalendarEntry::getDate() const {
    return date;
}

string CalendarEntry::getSubject() const {
    return subject;
}

bool CalendarEntry::getReminder() const {
    return reminder;
}

// Setter methods
void CalendarEntry::setDate(const Date& d) {
    date = d;
}

void CalendarEntry::setSubject(const string& s) {

```

```

        subject = s;
    }

    void CalendarEntry::setReminder(bool r) {
        reminder = r;
    }

    // Print the calendar entry details
    void CalendarEntry::printEntry() const {
        cout << "Date: ";
        date.printDate();
        cout << "Subject: " << subject << endl;
        cout << "Reminder: " << (reminder ? "On" : "Off") << endl;
    }

    // Ask the user for calendar entry details
    void CalendarEntry::askDetails() {
        cout << "Enter the calendar entry date: " << endl;
        date.askDate(); // Get the date from user

        cout << "Enter the subject of the entry: ";
        cin.ignore(); // Clear input buffer
        getline(cin, subject);

        // Ensure subject is not empty
        while (subject.empty()) {
            cout << "Subject cannot be empty. Please enter a valid subject: ";
            getline(cin, subject);
        }

        // Ask if the reminder is on
        char reminderChoice;
        do {
            cout << "Is the reminder on? (y/n): ";
            cin >> reminderChoice;
            reminderChoice = tolower(reminderChoice);
        } while (reminderChoice != 'y' && reminderChoice != 'n');

        reminder = (reminderChoice == 'y');
    }
}

```

4.1.5 calendareentry,h

```

#ifndef CALENDAREENTRY_H
#define CALENDAREENTRY_H

#include <iostream>
#include <string>
#include "Date.h"

```

```

using namespace std;

class CalendarEntry {
private:
    Date date;
    string subject;
    bool reminder;

public:
    // Constructors
    CalendarEntry();
    CalendarEntry(const Date& d, const string& s, bool r);
    CalendarEntry(const CalendarEntry& other); // Copy constructor

    // Destructor
    ~CalendarEntry();

    // Getters
    Date getDate() const;
    string getSubject() const;
    bool getReminder() const;

    // Setters
    void setDate(const Date& d);
    void setSubject(const string& s);
    void setReminder(bool r);

    // Methods
    void printEntry() const;
    void askDetails();
};

#endif

```

4.2 teht 2

4.2.1 main.cpp

```

#include <iostream>
#include "CalendarEntry.h"
#include "Date.h"

using namespace std;

void doSomethingByValue(CalendarEntry aCalendarEntry) {
    cout << "Do something with Calendar Entry (value parameter):" <<
endl;
    aCalendarEntry.printEntry(); // Copies the object's data and prints
it
}

```

```

void doSomethingByReference(CalendarEntry& aCalendarEntry) {
    cout << "Do something with Calendar Entry (reference parameter):" <<
endl;
    aCalendarEntry.printEntry(); // Modifies the original object di-
rectly
}

void doSomethingByConstReference(const CalendarEntry& aCalendarEntry) {
    cout << "Do something with Calendar Entry (const reference parame-
ter):" << endl;
    aCalendarEntry.printEntry(); // You can only call const methods
}

int main() {
    // Create a calendar entry with a specific date
    Date date(12, 4, 2024);
    CalendarEntry entry(date, "Meeting at 10 AM", true);

    doSomethingByValue(entry);

    doSomethingByReference(entry);

    // Show that original object may have been modified
    cout << "\nAfter reference parameter function call:" << endl;
    entry.printEntry();

    // Pass the object by const reference
    doSomethingByConstReference(entry);

    return 0;
}

```

4.2.2 date.cpp

```

#include "Date.h"

// Default constructor
Date::Date() {
    day = 1;
    month = 1;
    year = 2000;
}

// Parameterized constructor
Date::Date(int d, int m, int y) {
    day = d;
    month = m;
    year = y;
}

```

```

// Getter methods
int Date::getDay() const {
    return day;
}

int Date::getMonth() const {
    return month;
}

int Date::getYear() const {
    return year;
}

// Setter methods
void Date::setDay(int d) {
    day = d;
}

void Date::setMonth(int m) {
    month = m;
}

void Date::setYear(int y) {
    year = y;
}

// Print the date
void Date::printDate() const {
    cout << day << "/" << month << "/" << year << endl;
}

// Ask for date input
void Date::askDate() {
    cout << "Enter day: ";
    cin >> day;
    cout << "Enter month: ";
    cin >> month;
    cout << "Enter year: ";
    cin >> year;
}

```

4.2.3 date.h

```

#ifndef DATE_H
#define DATE_H

#include <iostream>
#include <string>

```

```

using namespace std;

class Date {
private:
    int day;
    int month;
    int year;

public:
    // Constructors
    Date(); // Default constructor
    Date(int d, int m, int y); // Parameterized constructor

    // Getter methods
    int getDay() const;
    int getMonth() const;
    int getYear() const;

    // Setter methods
    void setDay(int d);
    void setMonth(int m);
    void setYear(int y);

    void printDate() const;
    void askDate();
};

#endif

```

4.2.4 calendarentry.cpp

```

#include "CalendarEntry.h"

// Default constructor
CalendarEntry::CalendarEntry() {
    date = Date();
    subject = "Not defined";
    reminder = false;
}

// Parameterized constructor
CalendarEntry::CalendarEntry(Date d, string s, bool r) {
    date = d;
    subject = s;
    reminder = r;
}

// Destructor
CalendarEntry::~CalendarEntry() {
    cout << "Calendar entry deleted." << endl;
}

```

```

}

// Getter methods
Date CalendarEntry::getDate() const {
    return date;
}

string CalendarEntry::getSubject() const {
    return subject;
}

bool CalendarEntry::getReminder() const {
    return reminder;
}

// Setter methods (const-viite)
void CalendarEntry::setDate(const Date& d) {
    date = d;
}

void CalendarEntry::setSubject(const string& s) {
    subject = s;
}

void CalendarEntry::setReminder(bool r) {
    reminder = r;
}

// Print the calendar entry
void CalendarEntry::printEntry() const {
    cout << "Date: ";
    date.printDate();
    cout << "Subject: " << subject << endl;
    cout << "Reminder: " << (reminder ? "On" : "Off") << endl;
}

// Ask the user for calendar entry details
void CalendarEntry::askDetails() {
    cout << "Enter the calendar entry date: " << endl;
    date.askDate();
    cout << "Enter the subject of the entry: ";
    cin.ignore(); // Clears the input buffer
    getline(cin, subject);
    char reminderChoice;
    cout << "Is the reminder on? (y/n): ";
    cin >> reminderChoice;
    reminder = (reminderChoice == 'y' || reminderChoice == 'Y');
}

```


4.2.5 calendareentry.h

```
#ifndef CALENDAREENTRY_H
#define CALENDAREENTRY_H

#include <iostream>
#include <string>
#include "Date.h"

using namespace std;

class CalendarEntry {
private:
    Date date;
    string subject;
    bool reminder;

public:
    CalendarEntry();
    CalendarEntry(Date d, string s, bool r);
    ~CalendarEntry();

    Date getDate() const;
    string getSubject() const;
    bool getReminder() const;

    void setDate(const Date& d); // const
    void setSubject(const string& s); // const
    void setReminder(bool r);

    void printEntry() const;
    void askDetails();
};

#endif
```

4.3 teht3

4.3.1 main.cpp

```
#ifndef CALENDAREENTRY_H
#define CALENDAREENTRY_H

#include <iostream>
#include <string>
#include "Date.h"

using namespace std;

class CalendarEntry {
private:
```

```

    Date date;
    string subject;
    bool reminder;

public:
    CalendarEntry();
    CalendarEntry(Date d, string s, bool r);
    ~CalendarEntry();

    Date getDate() const;
    string getSubject() const;
    bool getReminder() const;

    void setDate(const Date& d); // const
    void setSubject(const string& s); // const
    void setReminder(bool r);

    void printEntry() const;
    void askDetails();
};

#endif

```

4.3.2 henkilo.cpp

```
#include "henkilo.h"
```

```
Henkilo::Henkilo(string n, int i, string o) : nimi(n), ika(i), osoite(o) {}
```

```

void poistaHenkilo(vector<Henkilo>& henkilot, const string& poistettavaNimi) {
    auto it = remove_if(henkilot.begin(), henkilot.end(),
        [&](const Henkilo& h) { return h.nimi == poistettavaNimi; });

    if (it != henkilot.end()) {
        henkilot.erase(it, henkilot.end());
        cout << "Henkilö " << poistettavaNimi << " poistettu.\n";
    } else {
        cout << "Henkilöä " << poistettavaNimi << " ei löytynyt.\n";
    }
}

```

4.3.3 henkilo.h

```
#ifndef HENKILO_H
#define HENKILO_H

#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

struct Henkilo {
    string nimi;
    int ika;
    string osoite;

    Henkilo(string n, int i, string o);
};

// Function to delete a person by name
void poistaHenkilo(vector<Henkilo>& henkilot, const string& poistettavaNimi);

#endif
```

4.4 teht4

4.4.1 main.cpp

```
#include "henkilo.h"
#include <vector>

int main() {
    vector<Henkilo*> henkilot;

    henkilot.push_back(new Henkilo("Anne", 20, "Katu 1, Helsinki"));
    henkilot.push_back(new Henkilo("Kalle", 30, "Tie 2, Tampere"));
    henkilot.push_back(new Henkilo("Matti", 25, "Kuja 3, Turku"));
    henkilot.push_back(new Henkilo("Liisa", 28, "Polku 4, Oulu"));
}
```

```

    henkilot.push_back(new Henkilo("Pekka", 35, "Raitti 5, Espoo"));

    cout << "\nTulostetaan henkilöt:\n";
    for (const auto& h : henkilot) {
        cout << h->nimi << ", " << h->ika << " vuotta, " << h->osoite <<
endl;
    }

    for (auto& h : henkilot) {
        delete h;
    }

    henkilot.clear();

    return 0;
}

```

4.4.2 henkilo.cpp

```

#include "henkilo.h"

// Normal constructor
Henkilo::Henkilo(string n, int i, string o) : nimi(n), ika(i), osoite(o)
{
    cout << "Luotiin Henkilo: " << nimi << ", " << ika << " vuotta, " <<
osoite << endl;
}

// Copy constructor
Henkilo::Henkilo(const Henkilo& h) : nimi(h.nimi), ika(h.ika),
osoite(h.osoite) {
    cout << "Kopioitiin Henkilo: " << nimi << ", " << ika << " vuotta, "
<< osoite << endl;
}

// Destructor
Henkilo::~Henkilo() {
    cout << "Poistetaan Henkilo: " << nimi << ", " << ika << " vuotta, "
<< osoite << endl;
}

```

4.4.3 henkilo.h

```

#ifndef HENKILO_H
#define HENKILO_H

#include <iostream>

```

```
#include <string>

using namespace std;

class Henkilo {
public:
    string nimi;
    int ika;
    string osoite;

    // Normaali konstruktori
    Henkilo(string n, int i, string o);

    // Kopiorakentaja
    Henkilo(const Henkilo& h);

    // Destruktori
    ~Henkilo();
};

#endif
```

5 Viikkotehtävät

6 Viikkotehtävät