Умова

Відповідно варіанту до тексту першої лабораторної роботи потрібно внести наступні зміни:
- визначити базовий та похідний класи;
- переписати оголошення похідного класу відповідно синтаксису для відкритого успадкування;
- переробити оголошення й реалізацію конструкторів похідного класу, у конструкторі з параметрами обов’язково передбачити використання конструктора з параметрами базового класу;
- у реалізації функції виведення на екран інформації в похідному класі використати відповідну функцію базового класу;
- до конструкторів та деструкторів базового, похідного та одного з агрегованих
класів
долучити
виведення
відповідної
контрольної
інформації для можливості відстеження порядку створення та знищення об’єктів.

Базовий – Замовник, похідний – Дослідження.

Текст програми

<u>lab1mod.hpp</u>

```
class Date{
   private:
      int day, month, year;
   public:
      Date();
       Date(int, int, int);
      Date(const Date&);
      int getDay() const;
      int getMonth() const;
      int getYear() const;
      char *out() const;
      Date& modDay(int);
      Date& modMonth(int);
      Date& modYear(int);
      ~Date();
      static bool verify(const int*);
};

class Student{
   private:
      char *name, *surname;
      int enroll_year;
   public:
      Student();
      Student(const char*, const char*, int);
      Student(const Student&);
      const char* getName() const;
      const char* getSurname() const;
      int getEnrollYear() const;
      char* out() const;
      Student& modName(const char *);
      Student& modSurname(const char *);
      Student& modEnrollYear(int);
      ~Student();
};

class Customer{
   protected:
      char *name, *theme;
      int price;
   public:
      Customer();
      Customer(const char*, const char*, int);
      Customer(const Customer&);
      const char* getName() const;
      const char* getTheme() const;
      int getPrice() const;
      char *out() const;
      Customer& modName(const char*);
      Customer& modTheme(const char*);
      Customer& modPrice(int);
      ~Customer();
};

enum SciAchivment {THESIS, ARTICLE, REPORT, INTARTICLE};

class Publication{
   private:
      Student* author;
      SciAchivment pub_type;
   public:
      Publication();
      Publication(const Student&, const SciAchivment);
      Publication(const Publication&);
      const Student& getAuthor() const;
      SciAchivment getPublicationType() const;
      char * out() const;
      Publication& modAuthor(const Student&);
      Publication& modPublicationType(const SciAchivment);
      ~Publication();
};
```

```
class Research: public Customer{
    private:
        Date* sign_date;
        Publication** publications;
        int num_of_publications;
    public:
        Research();
        Research(const Customer&, const Date&);
        Research(const Research&);
        const Date& getSignDate() const;
        int getNumOfPublications() const;
        const Publication* const* getPublicationList() const;
        char * getInfo() const;
        char * out() const;
        Research& addPublication(const Publication&);
        Research& modCustomer(const Customer&);
        ~Research();
};
```

lab1mod.cpp

```
#include "lab1mod.hpp"
#include <stdio.h>
#include <ctime>
#include <cmath>
#include <cstring>
#include <cstdlib>
#include <regex>
#include <iostream>
//For debugging
//#include <iostream>

//-------------------------------------------------

//CONSTRUCTORS DATE

Date::Date(){
    //std::time_t time( std::time_t* arg ) - returns the current calendar time
encoded as a std::time_t object, and also stores it
    //in the object pointed to by arg, unless arg is a null pointer.
    std::time_t now = std::time(NULL);
    //struct tm* - pointer on time structure; structure containing a calendar date
and time broken down into its components.
    //struct tm * localtime (const time_t * timer) - uses the value pointed by
timer to fill a tm structure with the values that
    //represent the corresponding time, expressed for the local timezone.
    struct tm* tstruct = std::localtime(&now);
    //and now aggign values from received structure-calendar
    this->day = tstruct->tm_mday;
    this->month = tstruct->tm_mon + 1;
    this->year = tstruct->tm_year + 1900;
    std::cout<<"-Date wurde default ERSTELLT!-\n";
};

Date::Date(int in_day, int in_month, int in_year){
    this->day = in_day;
    this->month = in_month;
    this->year = in_year;
    std::cout<<"-Date wurde custom ERSTELLT!-\n";
}

//constructor of copy
Date::Date(const Date& in_date){
    this->day = in_date.getDay(); //as fields day, month, year are private; we
can change 'em only with the class methods
    this->month = in_date.getMonth();
    this->year = in_date.getYear();
    std::cout<<"-Date wurde copy ERSTELLT!-\n";
}

//CONSTRUCTORS DATE

//DESTRUCTOR DATE
Date::~Date(){
    std::cout<<"--Date wurde KAPUTT!--\n";
}

int Date::getDay() const{
    return this->day;
}

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

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

//date output in string representation
char * Date::out() const{
    char * res = new char[10];
    sprintf(res, "%02i.%02i.%04i", this->day, this->month, this->year);
    return res;
}

//modifications of day, month, year
Date& Date::modDay(int inday){
    this->day = inday;
    return *this;
}

Date & Date::modMonth(int in_month){
    this->month = in_month;
    return *this;
}

Date& Date::modYear(int in_year){
    this->year = in_year;
    return *this;
}
//

//check whether the date is correct
bool Date::verify(const int * in_date){
    if (! (1582<= *(in_date + 2) )  )
        return false;
    if (! (1<= *(in_date + 1) && *(in_date + 1)<=12) )
        return false;
    if (! (1<= *(in_date) && *(in_date)<=31) )
        return false;
    if ( (*(in_date)==31) && (*(in_date + 1)==2 || *(in_date + 1)==4 || *(in_date
+ 1)==6 || *(in_date + 1)==9 || *(in_date + 1)==11) )
        return false;
    if ( (*(in_date)==30) && (*(in_date + 1)==2) )
        return false;
    if ( (*(in_date + 1)==2) && (*(in_date)==29) && (*(in_date + 2)%4!=0) )
        return false;
    if ( (*(in_date + 1)==2) && (*(in_date)==29) && (*(in_date + 2)%400==0) )
        return true;
    if ( (*(in_date + 1)==2) && (*(in_date)==29) && (*(in_date + 2)%100==0) )
        return false;
    if ( (*(in_date + 1)==2) && (*(in_date)==29) && (*(in_date + 2)%4==0)  )
        return true;
    return true;
}

//-------------------------------------------------

//-------------------------------------------------

//CONSTRUCTORS STUDENT
Student::Student(){
    this->name = new char[sizeof "Name"];
    this->surname = new char[sizeof "Surname"];
    std::strcpy(this->name, "Name");
    std::strcpy(this->surname, "Surname");
    //
    std:: time_t now = std::time(NULL);
    struct tm *tstruct = std::localtime(&now);
    //
    this->enroll_year = 1900 + tstruct->tm_year;
    return ;
}
```

```
Student::Student(const char * in_name, const char * in_surname, int in_year){
    this->name = new char[sizeof in_name];
    this->surname = new char[sizeof in_surname];
    std::strcpy(this->name, in_name);
    std::strcpy(this->surname, in_surname);
    this->enroll_year = in_year;
    return ;
}

//constructor of copy
Student::Student(const Student& in_student){
    this->name = new char[sizeof in_student.name];
    this->surname = new char[sizeof in_student.surname];
    std::strcpy(this->name, in_student.name);
    std::strcpy(this->surname, in_student.surname);
    this->enroll_year = in_student.enroll_year;
    return ;
}
//CONSTRUCTORS STUDENT

//DESTRUCTOR STUDENT
Student::~Student(){
    //free memory
    delete[] this->name;
    delete[] this->surname;
    return ;
}

const char * Student::getName() const{
    return (const char *)this->name;
}

const char * Student::getSurname() const{
    return (const char *)this->surname;
}

int Student::getEnrollYear() const{
    return this->enroll_year;
}

//output full info about student
char * Student::out() const{
    char * res = new char[(std::strlen(this->name) + 1) + (std::strlen(this-
>surname) + 1) + 6];
    sprintf(res, "%s %s\n%04i",
                    this->name, this->surname, this->enroll_year); //sends
formatted output to a string res
    return res;
}

//modife student's name
Student& Student::modName(const char * in_name){
    delete[] this->name;
    this->name = new char[sizeof in_name];
    std::strcpy(this->name, in_name);
    return *this;
}

Student& Student::modSurname(const char * in_surname){
    delete[] this->surname;
    this->surname = new char[sizeof in_surname];
    std::strcpy(this->surname, in_surname);
    return *this;
}

Student & Student::modEnrollYear(int inyear){
    this->enroll_year = inyear;
    return *this;
}

//-------------------------------------------------

//-------------------------------------------------

//CONSTRUCTORS CUSTOMER

Customer::Customer(){
    this->name = new char[sizeof "Name"];
    this->theme = new char[sizeof "Theme"];
    std::strcpy(this->name, "Name");
    std::strcpy(this->theme, "Theme");
    this->price = 0;
    std::cout<<"-Customer wurde default ERSTELLT!-\n";
}

Customer::Customer(const char * in_name, const char * in_theme, int in_price)
{
    this->name = new char[sizeof in_name];
    this->theme = new char[sizeof in_theme];
    std::strcpy(this->name, in_name);
    std::strcpy(this->theme, in_theme);
    this->price = in_price;
    std::cout<<"-Customer wurde custom ERSTELLT!-\n";
}

//constructor of copy
Customer::Customer(const Customer& in_research){
    this->name = new char[sizeof in_research.name];
    this->theme = new char[sizeof in_research.theme];
    std::strcpy(this->name, in_research.name);
    std::strcpy(this->theme, in_research.theme);
    this->price = in_research.price;
    std::cout<<"-Customer wurde copy ERSTELLT!-\n";
}
```

```
//CONSTRUCTORS CUSTOMER

//DESTRUCTOR CUSTOMER
Customer::~Customer(){
    delete[] this->name;
    delete[] this->theme;
    std::cout<<"--Customer wurde KAPUTT!--\n";
}

const char * Customer::getName() const{
    return (const char *)this->name;
}

const char * Customer::getTheme() const{
    return (const char *)this->theme;
}

int Customer::getPrice() const{
    return this->price;
}

//output full info about customer
char * Customer::out() const{
    char * res = new char[(std::strlen(this->name) + 1) + (std::strlen(this-
>theme) + 1) + 6];
    sprintf(res, "%s %s\n%04i",
                    this->name, this->theme, this->price);
    return res;
}

Customer & Customer::modName(const char * in_name){
    delete[] this->name;
    this->name = new char[sizeof in_name];
    std::strcpy(this->name, in_name);
    return *this;
}

Customer & Customer::modTheme(const char * in_theme){
    delete[] this->theme;
    this->theme = new char[sizeof in_theme];
    std::strcpy(this->theme, in_theme);
    return *this;
}

Customer & Customer::modPrice(int in_price){
    this->price = in_price;
    return *this;
}

//-----------------------------------------------

//-----------------------------------------------
```

```
//CONSTRUCTORS PUBLICATION

Publication::Publication(){
    this->author = new Student();
    this->pub_type = THESIS;
    std::cout<<"-Publication wurde default ERSTELLT!-\n";
};

Publication::Publication(const Student& in_author, const SciAchivment
in_pub_type){
    this->author = new Student(in_author);//initialize Student with some name
in_author
    this->pub_type = in_pub_type;
    std::cout<<"-Publication wurde custom ERSTELLT!-\n";
};

//constructor of copy
Publication::Publication(const Publication& in_publication){
    this->author = new Student(in_publication.getAuthor());
    this->pub_type = in_publication.getPublicationType();
    std::cout<<"-Publication wurde copy ERSTELLT!-\n";
};

//CONSTRUCTORS PUBLICATION

//get author
const Student& Publication::getAuthor() const{
    return *(this->author);
};

//get publication type
SciAchivment Publication::getPublicationType() const{
    return this->pub_type;
};

//output all info about Publication
char * Publication::out() const{
    char * a_name = this->author->out();
    char * p_name;
    switch(this->pub_type){
        case THESIS : p_name = new char[sizeof "thesis for report"];
                    std::strcpy(p_name, "thesis for report");
                    break;
        case ARTICLE : p_name = new char[sizeof "article in proffesional journal"];
                     std::strcpy(p_name, "article in proffesional journal");
                     break;
        case REPORT : p_name = new char[sizeof "report on conference"];
                    std::strcpy(p_name, "report on conference");
                    break;
        case INTARTICLE : p_name = new char[sizeof "article in an international
science journal"];
```

```
                        std::strcpy(p_name, "article in an international science
journal");
                        break;
    };
    char * res = new char[(std::strlen(a_name) + 1) + (std::strlen(p_name) + 1)
+ sizeof "\nPublication type : "];
    std::strcpy(res, a_name);
    std::strcat(res, "\tPublication type : ");
    std::strcat(res, p_name);
    delete[] p_name;
    delete[] a_name;
    return res;
};

//change author
Publication& Publication::modAuthor(const Student & in_author){
    delete this->author;
    this->author = new Student(in_author);
    return *this;
};

//change publication type
Publication& Publication::modPublicationType(const SciAchivment in_pub_type)
{
    this->pub_type = in_pub_type;
    return *this;
};

//DESTRUCTOR PUBLICATION
Publication::~Publication(){
    delete this->author;
    std::cout<<"--Publication wurde KAPUTT!--\n";
};

//------------------------------------------------

//------------------------------------------------

//CONSTRUCTORS RESEARCH

Research::Research(){
    this->sign_date = new Date();
    this->num_of_publications = 0;
    this->publications = NULL;
    std::cout<<"-Research wurde default ERSTELLT!-\n";
};

Research::Research(const Customer& in_customer, const Date&
in_date):Customer(in_customer){
    this->sign_date = new Date(in_date);
    this->num_of_publications = 0;
    this->publications = NULL;
    std::cout<<"-Research wurde custom ERSTELLT!-\n";
};

Research::Research(const Research& in_research){
    strcpy(this->name, in_research.name);
    strcpy(this->theme, in_research.theme);
    this->price = in_research.price;
    this->sign_date = new Date(in_research.getSignDate());
    this->num_of_publications = in_research.getNumOfPublications();
    this->publications = new Publication*[this->num_of_publications];
    const Publication * const * retireved = in_research.getPublicationList();
    for (int i=0; i++; i < this->num_of_publications){
        *(this->publications + i) = new Publication(**(retireved + i));
    };
    std::cout<<"-Research wurde copy ERSTELLT!-\n";
};

//CONSTRUCTORS RESEARCH


const Date& Research::getSignDate() const{
    return *(this->sign_date);
};

int Research::getNumOfPublications() const{
    return this->num_of_publications;
};

const Publication* const* Research::getPublicationList() const{
    return this->publications;
};

char* Research::getInfo() const{
    char* res = new char[(std::strlen(this->theme) + 1) +
                      sizeof "theme: \nnum of publications:" +
                      (sizeof (char))*(int)(std::log(this->num_of_publications ?
                                                  this->num_of_publications != 0 :
                                                  1)
                                              / std::log(10))];
    sprintf(res, "theme: %s\nnum of publications: %i",
                this->theme, this->num_of_publications);
    return res;
};

char* Research::out() const{
    //getting fields out-strings
    char* customer_out = Customer::out(); //using the base function out()
    char* date_out = this->sign_date->out();
    char* nop_out = new char[sizeof "\nNumber of all publications" +
                          (int)(1 + std::log((this->num_of_publications != 0 )? this-
>num_of_publications : 1) /
                                std::log(10))
```

```
                              ];
    //Formed number of all publications
    sprintf(nop_out, "%s:\t%i",
                 "\nNumber of all publications", this->num_of_publications);
    //getting data from all publications and calulating their overall size
    char ** pubs_out = new char*[num_of_publications];
    int totalsize = 0;
    for (int i=0; i < this->num_of_publications; i++){
        *(pubs_out + i) = (*(this->publications + i))->out();
        totalsize += std::strlen(*(pubs_out + i));
    };
    //allocating resulting string, with size as sum of all pieces
    char * res = new char[sizeof "Customer information:\t" +
(std::strlen(customer_out) + 1) +
                          sizeof "Signing date:\t" + (std::strlen(date_out) + 1) +
                          (std::strlen(nop_out) + 1) +
                          sizeof "\nList of all publications:\n" + totalsize +
                          (sizeof "\t")*num_of_publications
                          ];
    //Collecting all strings in resulting string
    std::strcpy(res, "Customer information:\t");
    std::strcat(res, customer_out);
    std::strcat(res, "\nSigning date:\t");
    std::strcat(res, date_out);
    std::strcat(res, nop_out);
    std::strcat(res, "\nList of all publications:\n");
    for (int i=0; i < num_of_publications; i++){
        std::strcat(res, "\t");
        std::strcat(res, *(pubs_out + i));
    };
    //now it's time to deallocate these arrays
    delete[] customer_out;
    delete[] date_out;
    delete[] nop_out;
    for (int i=0; i < num_of_publications; i++){
        delete[] *(pubs_out + i);
    };
    delete [] pubs_out;
    return res;
};

Research& Research::addPublication(const Publication& in_publication){
    this->num_of_publications += 1;
    this->publications = (Publication**)std::realloc(this->publications, this-
>num_of_publications * sizeof (Publication*));
    *(this->publications + this->num_of_publications - 1) = new
Publication(in_publication);
    return *this;
};

Research& Research::modCustomer(const Customer& in_customer){
    this->modName(in_customer.getName());
    this->modTheme(in_customer.getTheme());
    this->modPrice(in_customer.getPrice());
    return *this;
};

//DESTRUCTOR RESEARCH
Research::~Research(){
    for (int i=0; i < num_of_publications; i++){
        delete *(this->publications + i);
    };
    std::free(this->publications);
    std::cout<<"--Research wurde KAPUTT!--\n";
};
```

lab1.cpp

```
#include "lab1mod.hpp"
#include <iostream>
#include <regex>

#define NUM_OF_OBJECTS 3

//char to date
int* char2date(const char * in_string){
    std::regex date_regex = std::regex("^([[:digit:]]{1,2})\\.([[:digit:]]{1,2})\\.
([[:digit:]]{1,4}$)");
    std::cmatch m;
    if (std::regex_search(in_string, m, date_regex)) {
        int * res = new int[3];
        for (int i=0; i < 3; i++)
            res[i] = std::atoi(m[i+1].str().c_str());
        return res;
    } else {
        std::cerr << "What's a pity, couldn't fit format." << std::endl;
        return NULL;
        };
};

int* return_date(char* oooh){
        return char2date(oooh);;
}

int main(){
    //create objects
    Date* date_for_example;
    Customer* customer_for_example;
    Publication *publication_example;
    Research* example[NUM_OF_OBJECTS]; //array of researches
    char* res;

    //enter date for research
    std::cout << "Please enter date in day.month.year format and then press
Enter." << std::endl;
    char* buf = new char[8];
    std::cin >> buf;
    int* date_for_example_as_int = char2date(buf);
    while (date_for_example_as_int == NULL){
        std::cout << "Please enter date in day.month.year format and then press
Enter." << std::endl;
        std::cin >> buf;
        date_for_example_as_int=return_date(buf);
    }
    delete buf;
    //after input we check whether the date is valid
    if (Date::verify(date_for_example_as_int)) {
        date_for_example = new Date(*(date_for_example_as_int),
                                    *(date_for_example_as_int + 1),
                                    *(date_for_example_as_int + 2));
        delete date_for_example_as_int;
    } else {
        std::cerr << "Sorry, date isn't valid... C u!" << std::endl;
        delete date_for_example_as_int;
        return 0;
          };

    //enter customer data for research
    char* name_buf = new char[80];
    char* theme_buf = new char[80];
    int price;
    std::cout << "Please enter customer's name (ascii only, not more than 80
charachters)." << std::endl;
    std::cin >> name_buf;
    std::cout << "Please enter theme (ascii only, not more than 80 charachters)
and then press Enter." << std::endl;
    std::cin >> theme_buf;
    std::cout << "Please enter price (int only) and then press Enter." <<
std::endl;
    std::cin >> price;
    customer_for_example = new Customer(name_buf, theme_buf,
price);//cause in our case we declared customer_for_example as a pointer

    example[0] = new Research(); //standart constructor(everything is
predefined)
    example[1] = new Research(*customer_for_example,
*date_for_example);//our "input" constructor
    example[2] = new Research(*example[1]);//constructor of copy

    //enter data for publication

    for (int i=0; i < NUM_OF_OBJECTS; i++){
        std::cout << "\n------------Example " << i+1 << "------------"<<std::endl;

        res = example[i]->getInfo();
        std::cout <<"\nShort info:" << std::endl << res << std::endl ;
        delete res;
        char* surname_temp = new char[80];
        std::cout<<"\nInput student's surname: ";
        std::cin>>surname_temp;
        std::cout<<" "<<std::endl;;
        Student stud_example (name_buf, surname_temp, 2016);
        publication_example = new Publication(stud_example, ARTICLE);
        example[i]->addPublication( *publication_example);
        res = example[i]->out();
        std::cout << "\nFull-size out:" << std::endl << res << std::endl ;
        res = example[i]->getInfo();
        std::cout <<"\nShort info after adding a publication:" << std::endl << res
<< std::endl ;

        delete example[i];
        delete publication_example;
    };
    //delete example;
    delete date_for_example;
    delete customer_for_example;
    return 0;
};
```

Результати роботи програми

```
alextr@alextr:/media/alextr/DATA/5 semester/OOP (C++,
C#)/Labwork2$ make lab1
g++ -c -std=c++11 -o obj/lab1.o lab1.cpp
g++ -c -std=c++11 -o obj/lab1mod.o lab1mod.cpp
g++ -o lab1 obj/lab1.o obj/lab1mod.o lab1mod.hpp
alextr@alextr:/media/alextr/DATA/5 semester/OOP (C++,
C#)/Labwork2$ ./lab1
alextr@alextr:/media/alextr/DATA/5 semester/OOP (C++, C#)/Labwork2$ ./lab1
Please enter date in day.month.year format and then press Enter.
12.12.1997
-Date wurde custom ERSTELLT!-
Please enter customer's name (ascii only, not more than 80 charachters).
Andrew
Please enter theme (ascii only, not more than 80 charachters) and then press
Enter.
Programming
Please enter price (int only) and then press Enter.
124
-Customer wurde custom ERSTELLT!-
-Customer wurde default ERSTELLT!-
-Date wurde default ERSTELLT!-
-Research wurde default ERSTELLT!-
-Customer wurde copy ERSTELLT!-
-Date wurde copy ERSTELLT!-
-Research wurde custom ERSTELLT!-
```

```
-Customer wurde default ERSTELLT!-
-Date wurde copy ERSTELLT!-
-Research wurde copy ERSTELLT!-

------------Example 1------------

Short info:
theme: Theme
num of publications: 0

Input student's surname: Kurylenko

-Publication wurde custom ERSTELLT!-
-Publication wurde copy ERSTELLT!-

Full-size out:
Customer information:  Name Theme
0000
Signing date:       07.11.2016
Number of all publications:    1
List of all publications:
        Andrew Kurylenko
2016    Publication type : article in proffesional journal

Short info after adding a publication:
theme: Theme
num of publications: 1
--Publication wurde KAPUTT!--
--Research wurde KAPUTT!--
--Customer wurde KAPUTT!--
--Publication wurde KAPUTT!--

------------Example 2------------

Short info:
theme: Programming
num of publications: 0

Input student's surname: Kozak

-Publication wurde custom ERSTELLT!-
-Publication wurde copy ERSTELLT!-

Full-size out:
Customer information:  Andrew Programming
0124
Signing date:       12.12.1997
Number of all publications:    1
List of all publications:
        Andrew Kozak
2016    Publication type : article in proffesional journal
```

```
Short info after adding a publication:
theme: Programming
num of publications: 1
--Publication wurde KAPUTT!--
--Research wurde KAPUTT!--
--Customer wurde KAPUTT!--
--Publication wurde KAPUTT!--

------------Example 3------------

Short info:
theme: Programming
num of publications: 0

Input student's surname: Kononenko

-Publication wurde custom ERSTELLT!-
-Publication wurde copy ERSTELLT!-

Full-size out:
Customer information:  Andrew Programming
0124
Signing date:       12.12.1997
Number of all publications:    1
List of all publications:
        Andrew Kononenko
2016  Publication type : article in proffesional journal

Short info after adding a publication:
theme: Programming
num of publications: 1
--Publication wurde KAPUTT!--
--Research wurde KAPUTT!--
--Customer wurde KAPUTT!--
--Publication wurde KAPUTT!--
--Date wurde KAPUTT!--
--Customer wurde KAPUTT!--
```