**Homework 2**

|  |
| --- |
| 그림입니다. 원본 그림의 이름: YU_UI_RGB-10.png 원본 그림의 크기: 가로 2256pixel, 세로 3047pixel 프로그램 이름 : Adobe ImageReady |

|  |  |
| --- | --- |
| 과목명 | 객체지향프로그래밍과자료구조 |
| 교수님 | 김영탁 교수님 |
| 이 름 | 김주환 |
| 학 번 | 21812158 |
| 일 자 | 2021.09.15.수 |

**2.1 Object-oriented programming**

|  |
| --- |
| **(1) C++ Object-Oriented programming에서 encapsulation이란 무슨 의미인가? Encapsulation을 함으로써 어떤 장점이 있는가?**   * 개요 데이터와 멤버함수를 하나의 캡슐로 만들어 사용하는 것 (멤버함수는 데이터가 항성 정상적인 범위 내에서 유지될 수 있도록 관리) * 항목 멤버 데이터 : 사용자 정보를 포함, 각 데이터에는 정상적인 범위가 지정되어 있음 멤버 함수 : 데이터를 처리하는 기능을 구현 * 예시 사람의 나이는 보통 0 ~ 150 살이다. 하지만, 프로그램을 사용할 경우 항상 정상적인 범위의 값이 입력되지는 않는다. 이 때 캡슐화를 통해 멤버 함수가 데이터를 정상 범위 내에 있도록 관리할 수 있다. * 특징 캡슐화를 통해 데이터 추상화 및 정보 은닉, 정보 보호가 가능해진다. 이는 후술하겠다. 데이터를 구별성 있게, 안전하게 관리 가능하다. |
| **(2) C++ Object-Oriented programming에서 information hiding이란 무슨 의미인가? 무엇을 누구에게 숨기는 것인가?**   * 개요 클래스 내부에서 데이터를 처리하는 기능이 어떻게 구현되었는가에 대한 상세한 정보는 제공하지 않고, 단지 사용 방법에 대한 추상적인 정보만 제공하는 것 클래스 내부에서 데이터를 처리하는 기능을 구현하는 방법은 새로운 기술이 개발됨에 따라 계속 바뀔 수 있으며, 따라서 구현 기술에 대한 상세한 내용은 사용자에게 알려주지 않고, 은닉시킴 새로운 기술의 도입을 쉽게 할 수 있게 하며, 새로운 기술이 도입되어도 사용하는 방법에서는 변경이 없도록 함 외부 접속과 구현을 분리시킴으로써, 접속은 유지한 채, 내부적인 구현 방법만 변경할 수 있게 함 새로운 기술을 쉽게 도입할 수 있는 장점을 제공함 * 예시 자동차의 엔진 룸 내부가 어떻게 구현되어 있고, 어떻게 동작하는가에 대한 정보 없이도, 자동차 운전석에 있는 기기와 장치를 사용하여 운전할 수 있다. 이는 외부 접속과 내부 구현은 분리해 놓았기 때문에 가능한 것이다. 이를 통해 새로운 기술이 적용되어도 사용자들은 금방 적응할 수 있는 것이다. 차량의 발전에 있어서 엔진과 같은 구동 방식은 공개하지 않고 은닉시킨다. 기술의 발전에 따라 가솔린, 디젤, 전기모터, 수소연료 방식으로 구현 기술이 바뀌어도 새로운 기술을 쉽게 도입할 수 있다. * 정리 하나의 기능은 다양한 기술로 구현될 수 있다. 다른 방식으로 구현된 기능도, 실제 이를 사용하게되는 접속 기능은 동일하다. 내부 모듈을 시대에 따라 교체하며 더 저렴한 비용으로 구현 가능하다. 구현 기술에 대한 상세한 내용을 사용자에게 알려주지 않고, 은닉시킨다. |
| **(3) C++ Object-oriented programming에서 data protection이란 무슨 의미인가? 어떻게 데이터를 보호하는가?**   * 개요 클래스 내부의 데이터를 외부 사용자가 직접 접속할 수 있도록 허용하지 않으며, 항상 외부로 공개된 멤버함수를 통하여 접속할 수 있도록 관리 정보가 비정상적으로 변경되는 것을 방지하며, 보호함 * 방법 Public 그리고 Private Members 구분을 통해 정보를 보호할 수 있다. Public member function은 사용자들이 사용할 수 있도록 공개된다. Private Data Member는 외부로 공개된 Public member function을 통해 접근될 수 있게 함으로써 정보를 보호하고, 내부 데이터가 항상 정상적인 범위로 유지될 수 있게 한다. * 예시 카카오맵 같은 경우에는 API가 사용자들에게 주어진다. 이를 통해서만 사용자들이 데이터에 접근할 수 있다. |
| **(4) C++ Object-oriented programming에서 information hiding과 data protection의 차이점은 무엇인가?**  information hiding과 data protection은 유사하다고 생각할 수 있다. 하지만, 실상은 다르다.   * information hiding 클래스 내부에서 데이터를 처리하는 기능이 어떻게 구현되었는가에 대한 상세한 정보는 제공하지 않고, 단지 사용 방법에 대한 추상적인 정보만 제공하는 것 클래스 내부에서 데이터를 처리하는 기능을 구현하는 방법은 새로운 기술이 개발됨에 따라 계속 바뀔 수 있으며, 따라서 구현 기술에 대한 상세한 내용은 사용자에게 알려주지 않고, 은닉시킴 새로운 기술의 도입을 쉽게 할 수 있게 하며, 새로운 기술이 도입되어도 사용하는 방법에서는 변경이 없도록 함 외부 접속과 구현을 분리시킴으로써, 접속은 유지한 채, 내부적인 구현 방법만 변경할 수 있게 함 새로운 기술을 쉽게 도입할 수 있는 장점을 제공함 * data protection 클래스 내부의 데이터를 외부 사용자가 직접 접속할 수 있도록 허용하지 않으며, 항상 외부로 공개된 멤버함수를 통하여 접속할 수 있도록 관리한다. 정보가 비정상적으로 변경되는 것을 방지하며, 보호한다. |

**2.2 Programming with class Date**

|  |
| --- |
| /\* main.cpp \*/  /\* Description  \* Date 객체를 생성한다.  \* 원하는 날짜를 입력하고 크리스마스까지 얼마나 남았는지 확인한다.  \* Programmed by J. H. Kim  \* Last updated : 2021-09-15 \*/  #include <iostream>  #include "Date.h"  using namespace std;  int main() {  Date AD010101(1, 1, 1), newYearDay(2021, 1, 1), today, christmas(2021, 12, 25); // 객체 생성  int year;  int daysToChristmas;  cout << "AD Jan. 1, 1 is ";  AD010101.print(); // 서기 1년 1월 1일 정보 출력  cout << endl;  cout << "New year's day of 2020 is ";  newYearDay.print(); // 금년 1월 1일 정보 출력  cout << endl;  cout << "Christmas of this year is ";  christmas.print(); // 크리스마스 정보 출력  cout << endl;  today.input(); // 오늘 정보 입력  cout << "Today is ";  today.print(); // 오늘 정보 출력  cout << endl;  if (today.getMonth() == christmas.getMonth() && // 입력받은 Date가 크리스마스인지 확인  today.getDay() == christmas.getDay()) {  cout << "Happy Christmas !!\n";  }  else { // 아닐경우  cout << "Sorry, today is not Christmas!";  daysToChristmas = christmas.getElapsedDays() - today.getElapsedDays(); // 크리스마스까지 얼마나 남았는지 계산  cout << " You must wait " << daysToChristmas << " day(s) to Christmas !" << endl;  }  return 0;  } |
| /\* Date.h \*/  #ifndef DATE\_H  #define DATE\_H  #include <iostream>  #include <cstdlib>  using namespace std;  #define WEEKDAT\_AD01JAN01 MON  #define DAYS\_PER\_WEEK 7  #define Secs\_in\_Minute 60  #define Secs\_in\_Hour (Secs\_in\_Minute \* 60)  #define Secs\_in\_DAY (Secs\_in\_Hour \* 24)  #define LOCAL\_GMT\_OFFSET\_HOUR 9  class Date {  public:  Date(); // 생성자  Date(int y, int m, int d); // 지정 생성  ~Date(); // 소멸자  void input(); // 입력  void print(); // 출력  void setDate(int y, int m, int dy); // 정보 설정  void setRandDateAttributes(); // 랜덤 정보 설정  void setMonth(int m); // 월 정보 설정  void setYear(int y); // 연 정보 설정  int getYear() { return year; } // 연 정보 반환  int getMonth() { return month; } // 월 정보 반환  int getDay() { return day; } // 일 정보 반환  int getYearDay(); // 경과 일 수  int getYearDay(int m, int d);  int getWeekDay(); // 요일 반환  int getElapsedDays(); // 최초부터 경과 일 수  int getElapsedDays(Date); // 최초부터 경과 일 수  void fprintCalendar\_Month(ostream& fout);  private:  bool isLeapYear(); // 윤년 확인  bool isLeapYear(int y); // 윤년 지정 확인  bool isValidDate(int y, int m, int d); // 유효한 날짜인지 확인  int year, month, day;  };  #endif |
| /\* Date.cpp \*/  #include <iostream>  #include <string>  #include <iomanip>  #include "Date.h"  enum WEEKDAY { SUN, MON, TUE, WED, THR, FRI, SAT };  enum MONTH { JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC, NUM\_MONTHS };  const char\* weekDayName[DAYS\_PER\_WEEK] = { "Sunday", "Monday", "Tuesday", "Wednesday",  "Thursday", "Friday", "Saturday" };  const char\* weekDayNameShort[DAYS\_PER\_WEEK] = { "SUN", "MON", "TUE", "WED", "THR", "FRI", "SAT" };  const char\* monthName[13] = { "", "January", "February", "March", "April", "May", "June", "July",  "August", "September", "October", "November", "December" };  Date::Date() {  year = 0, month = 0, day = 0;  }  Date::Date(int y, int m, int d) {  year = y, month = m, day = d;  }  Date::~Date() {  // cout << "Date object instance is destructed" << endl;  }  bool Date::isValidDate(int y, int m, int d) {  int days\_month[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };  if (isLeapYear(y)) days\_month[2] = 29; // 윤년 확인  if ((m >= 1) && (m <= 12) && (d >= 1) && (d <= days\_month[m])) return true; // 범위 내의 Date인지 확인  else {  cout << "Illegal date! (" << m << ", " << d << ") ==> Program aborted." << endl;  return false;  }  }  void Date::setDate(int y, int m, int d) {  if (isValidDate(y, m, d)) year = y, month = m, day = d;  else {  cout << "Invalid date (" << y << ", " << m << ", " << d << ")";  cout << "Program aborted !!" << endl;  exit(1);  }  }  void Date::setRandDateAttributes() {  int days\_month[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };  year = rand() % 2000 + 1000;  month = rand() % 12 + 1;  if (isLeapYear(year) && month == 2) days\_month[2] = 29; // 윤년 확인  day = rand() % days\_month[month] + 1;  }  void Date::setMonth(int newMonth) {  if ((newMonth >= 1) && (newMonth <= 12)) month = newMonth;  else {  cout << "Illegal month value! Program aborted." << endl;  exit(1);  }  day = 1;  }  void Date::setYear(int y) {  year = y;  }  int Date::getYearDay() {  int yearDay = 0;  getYearDay(this->month, this->day);  return yearDay;  }  int Date::getYearDay(int month, int day) {  int days\_month[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };  int yearDay = 0;  if (isLeapYear(this->year))  days\_month[2] = 29;  for (int i = 1; i < month; i++)  yearDay += days\_month[i];  return yearDay + day;  }  int Date::getElapsedDays() {  int elpsDay = 0;  elpsDay = getElapsedDays(\*this);  return elpsDay;  }  int Date::getElapsedDays(Date d) {  int yearDay;  int elpsDay = 0;  for (int y = 1; y < d.year; y++) {  if (isLeapYear(y)) elpsDay += 366;  else elpsDay += 365;  }  yearDay = getYearDay(d.month, d.day);  elpsDay += yearDay;  return elpsDay;  }  int Date::getWeekDay() {  int weekDay\_AD010101 = MON;  int weekDay;  int elapsedDays = 0;  elapsedDays = getElapsedDays();  weekDay = (elapsedDays + weekDay\_AD010101 - 1) % 7; // 최초는 월욜이면 추가 안하면 대지않나?  // cout << ", Elapsed days from AD Jan. 1, 1 (" << elapsedDays << ")";  return weekDay;  }  void Date::input() {  int y, m, d;  cout << "Enter date in year month day : ";  cin >> y >> m >> d;  if (isValidDate(y, m, d)) year = y, month = m, day = d;  else {  cout << "Illegal date! Program aborted." << endl;  exit(1);  }  }  bool Date::isLeapYear(int y) {  if ((y % 4 == 0 && y % 100 != 0) || y % 400 == 0)  return true;  else  return false;  }  bool Date::isLeapYear() {  return isLeapYear(year);  }  void Date::print() {  const char\* weekDayName[7] = { "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",  "Friday", "Saturday" };  const char\* monthName[13] = { "", "January", "February", "March", "April", "May", "June", "July",  "August", "September", "October", "November", "December" };  int yearDay = 0;  int weekDay;  if ((month >= 1) && (month <= 12))  cout << setw(10) << string(monthName[month]);  cout << " " << setw(2) << day << ", " << setw(4) << year;  yearDay = getYearDay();  weekDay = getWeekDay();  cout << " (" << setw(10) << string(weekDayName[weekDay]) << ")";  } |

텍스트이(가) 표시된 사진

자동 생성된 설명