**Lab 5**

|  |
| --- |
| 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 | 2021.10.01.금 |

|  |
| --- |
| /\* main.cpp \*/  /\* Description  \* 학생 클래스와 상속  \* Programmed by J. H. Kim  \* Last updated : 2021-10-01 \*/  #include <iostream>  #include <fstream>  #include "StudentArray.h"  #include <string>  #define NUM\_STUDENTS 10  void main() {  StudentArray studentArray(NUM\_STUDENTS);  Student st;  ofstream fout;  int st\_ids[NUM\_STUDENTS];  fout.open("output.txt");  if (fout.fail()) {  cout << "Fail to open an output file (output.txt)" << endl;  exit(1);  }  genST\_ids(NUM\_STUDENTS, st\_ids);  fout << "Initializing student array (num students : " << NUM\_STUDENTS << ")" << endl;  for (int i = 0; i < NUM\_STUDENTS; i++) {  st = genRandStudent(st\_ids[i]);  studentArray[i] = st;  }  fout << studentArray;  // 학번 정렬  fout << endl << "Sorting studentArray by student id : " << endl;  studentArray.sortByST\_ID();  fout << studentArray;  // 이름 정렬  fout << endl << "Sorting studentArray by student name : " << endl;  studentArray.sortByName();  fout << studentArray;  // 학점 정렬  fout << endl << "Sorting studentArray by GPA : " << endl;  studentArray.sortByGPA();  fout << studentArray;  // 생일 정렬  fout << endl << "Sorting studentArray by BirthDate : " << endl;  studentArray.sortByBirthDate();  fout << studentArray;  fout << endl;  fout.close();  } |
| /\* Date.h \*/  #ifndef D  #define D  #include <fstream>  using namespace std;  class Date {  friend istream& operator>>(istream&, Date&);  friend ostream& operator<<(ostream&, const Date&);  public:  Date(); // 생성자  Date(int y, int m, int d); // 생성자  void setDate(int newYear, int newMonth, int newDay); // Date 설정  int getYear() { return year; } // 연도 반환  int getYearDay(); // 이번년도 지난 일 수 반환  int getWeekDay(); // 요일 반환  int getElapsedDays(); // 최초부터 지난 일 수 반환  const Date operator=(const Date rightSide); // 대입연산자  bool operator>(Date rightSide); // 비교연산자  bool operator<(Date rightSide); // 비교연산자  bool operator==(Date rightSide); // 비교연산자  bool operator!=(Date rightSide); // 비교연산자  bool isLeapYear(int y); // 윤년 확인  private:  bool isLeapYear(); // 윤년 확인  bool isValidDate(int y, int m, int d); // 유효한 Date 인지 확인  int year;  int month;  int day;  };  bool isLeapYear(int y); // 윤년 확인  Date genRandDate(); // Date 랜덤 생성  #endif // !D |
| /\* Date.cpp \*/  #include <iostream>  #include <iomanip>  #include "Date.h"  istream& operator>>(istream& fin, Date& d) {  fin >> d.year >> d.month >> d.day;  return fin;  }  ostream& operator<<(ostream& fout, const Date& d) {  fout << "Birth date : (" << d.year << "-" << setw(2) << d.month << "-" << setw(2) << d.day << ")";  return fout;  }  Date::Date() {  year = 0, month = 0, day = 0;  }  Date::Date(int y, int m, int d) {  year = y, month = m, day = d;  }  void Date::setDate(int newYear, int newMonth, int newDay) {  if (isValidDate(newYear, newMonth, newDay))  year = newYear, month = newMonth, day = newDay;  else {  cout << "Invalid date (" << newYear << ", " << newMonth << ", " << newDay << ")";  cout << "Program aborted !!" << endl;  exit(1);  }  }  int Date::getYearDay() {  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;  return yearDay;  }  int Date::getWeekDay() {  int weekDay\_AD010101 = 1;  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;  }  int Date::getElapsedDays() {  int yearDay;  int elpsDay = 0;  for (int y = 1; y < this->year; y++) {  if (isLeapYear(y)) elpsDay += 366;  else elpsDay += 365;  }  yearDay = getYearDay();  elpsDay += yearDay;  return elpsDay;  }  const Date Date::operator=(const Date rightSide) {  year = rightSide.year;  month = rightSide.month;  day = rightSide.day;  return \*this;  }  bool Date::operator>(Date rightSide) {  int tl, tr;  tl = getElapsedDays();  tr = rightSide.getElapsedDays();  if (tl > tr) return true;  else return false;  }  bool Date::operator<(Date rightSide) {  int tl, tr;  tl = getElapsedDays();  tr = rightSide.getElapsedDays();  if (tl < tr) return true;  else return false;  }  bool Date::operator==(Date rightSide) {  int tl, tr;  tl = getElapsedDays();  tr = rightSide.getElapsedDays();  if (tl == tr) return true;  else return false;  }  bool Date::operator!=(Date rightSide) {  int tl, tr;  tl = getElapsedDays();  tr = rightSide.getElapsedDays();  if (tl != tr) return true;  else return false;  }  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);  }  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;  if ((m >= 1) && (m <= 12) && (d >= 1) && (d <= days\_month[m])) return true; // 범위 내의 정보인지 확인  else {  cout << "Illegal date! (" << m << ", " << d << ") ==> Program aborted." << endl;  return false;  }  }  bool isLeapYear(int y) {  if ((y % 4 == 0 && y % 100 != 0) || y % 400 == 0)  return true;  else  return false;  }  Date genRandDate() {  Date d;  int days\_month[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };  int year, month, day;  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;  d.setDate(year, month, day);  return d;  } |
| /\* MyString.h \*/  #ifndef MS  #define MS  #include <string>  using namespace std;  string genRandName(); // 이름 랜덤 생성  string genRandDeptName(); // 학과 이름 랜덤 생성  #endif // !MS |
| /\* MyString.cpp \*/  #include <string>  #include "MyString.h"  using namespace std;  string genRandName() {  char str[8];  int name\_len, i = 0;  name\_len = rand() % (7 - 4) + 4;  str[0] = rand() % 26 + 'A';  for (i = 1; i < name\_len; i++)  str[i] = rand() % 26 + 'a';  str[i] = '\0';  return string(str);  }  string genRandDeptName() {  char str[5];  int name\_len, i = 0;  name\_len = rand() % (4 - 3) + 3;  str[0] = rand() % 26 + 'A';  for (i = 1; i < name\_len; i++)  str[i] = rand() % 26 + 'a';  str[i] = '\0';  return string(str);  } |
| /\* Person.h \*/  #ifndef P  #define P  #include <string>  #include "Date.h"  using namespace std;  class Person {  friend ostream& operator<<(ostream&, const Person&);  public:  Person() { birthDate = Date(0, 0, 0); name = ""; } // 생성자  Person(string nm, Date bd) { birthDate = bd; name = nm; } // 생성자  void setName(string n) { name = n; } // 이름 설정  void setBirthDate(Date bd) { birthDate = bd; } // 생일 설정  string getName() const { return name; } // 이름 반환  Date getBirthDate() const { return birthDate; } // 생일 반환  const Person& operator=(const Person&); // 대입연산자  protected:  Date birthDate;  string name;  };  #endif // !P |
| /\* Person.cpp \*/  #include <fstream>  #include <iomanip>  #include <string>  #include "Person.h"  #include "Date.h"  using namespace std;  ostream& operator<<(ostream& fout, const Person& p) {  fout << " Person [name: ";  fout.setf(ios::left);  fout << setw(16) << p.name;  fout << ", birth date: ";  fout.unsetf(ios::left);  fout << p.birthDate;  fout << "]";  return fout;  }  const Person& Person::operator=(const Person& right) {  name = right.name;  birthDate = right.birthDate;  return \*this;  } |
| /\* Student.h \*/  #ifndef S  #define S  #include "Person.h"  class StudentArray;  class Student : public Person {  friend class StudentArray;  friend ostream& operator<<(ostream&, const Student&);  public:  Student(); // 생성자  Student(int id); // 생성자  Student(int id, string n, Date dob, string dept\_n, double gpa); // 생성자  int getST\_id() const { return st\_id; } // 학번 반환  string getDept\_name() const { return dept\_name; } // 학과 이름 반환  double getGPA() const { return gpa; } // 학점 반환  Date getBirthDate() const { return birthDate; } // 생일 반환  void setST\_id(int id) { st\_id = id; } // 학번 설정  void setDept\_name(string dp\_n) { dept\_name = dp\_n; } // 학과 이름 설정  void setGPA(double g) { gpa = g; } // 학점 설정  const Student& operator=(const Student& right); // 대입연산자  bool operator>(const Student& right); // 비교연산자  bool operator==(const Student& right); // 비교연산자  private:  int st\_id;  string dept\_name;  double gpa;  };  Student genRandStudent(int id); // 학생 랜덤 생성  void genST\_ids(int num\_students, int\* st\_ids); // 학번 랜덤 생성  #endif // !S |
| /\* Student.cpp \*/  #include <iomanip>  #include "Student.h"  #include "MyString.h"  #include "Date.h"  using namespace std;  ostream& operator<<(ostream& fout, const Student& s) {  int count = 0;  fout.setf(ios::fixed);  fout.setf(ios::showpoint);  fout.precision(2);  fout << "Student(ID : " << setw(4) << s.st\_id;  fout << ", Name : " << setw(16) << s.name;  fout << ", Dept : " << setw(6) << s.dept\_name;  fout << ", birth date : " << setw(13) << s.birthDate;  fout << ", Grade : " << s.gpa;  return fout;  }  Student::Student()  : Person()  {  st\_id = 0;  dept\_name = "";  gpa = 0.0;  }  Student::Student(int id)  : Person()  {  st\_id = id;  dept\_name = "";  gpa = 0.0;  }  Student::Student(int id, string n, Date dob, string dept\_n, double gpa)  : Person(n, dob), st\_id(id), dept\_name(dept\_n), gpa(gpa)  {    }  const Student& Student::operator=(const Student& right) {  st\_id = right.st\_id;  dept\_name = right.dept\_name;  gpa = right.gpa;  // name = right.name;  // birthDate = right.birthDate;  Person::operator=(right);  return \*this;  }  bool Student::operator>(const Student& right) {  bool temp = true;  if (st\_id <= right.st\_id) temp = false;  if (dept\_name <= right.dept\_name) temp = false;  if (gpa <= right.gpa) temp = false;  if (name <= right.name) temp = false;  if (birthDate > right.birthDate);  else temp = false;  return temp;  }  bool Student::operator==(const Student& right) {  bool temp = true;  if (st\_id == right.st\_id) temp = false;  if (dept\_name == right.dept\_name) temp = false;  if (gpa == right.gpa) temp = false;  if (name == right.name) temp = false;  if (birthDate == right.birthDate) temp = false;  return temp;  }  Student genRandStudent(int id) {  Student st(id);  Date d;  double f, b;  st.setBirthDate(genRandDate());  st.setName(genRandName());  st.setDept\_name(genRandDeptName());  f = double(rand() % 100);  b = double(rand() % 100);  b /= 100.0;  st.setGPA(f + b);  return st;  }  void genST\_ids(int num\_students, int\* st\_ids) {  int\* flag = NULL;  int count = 0;  unsigned int u\_int32 = 0;  int bigRand;  flag = new int[40000];  for (int i = 0; i < num\_students; i++) {  flag[i] = 0;  }  while (count < num\_students) {  u\_int32 = ((long)rand() << 15) | rand(); // 비트이동으로 bigRand 생성  bigRand = u\_int32 % 40000 + 10000; // 원하는 크기로 스케일링  if (flag[bigRand - 10000] == 1) continue; // 중복 확인  flag[bigRand - 10000] = 1; // 중복 체크  st\_ids[count++] = bigRand;  }  delete[]flag;  } |
| /\* StudentArray.h \*/  #ifndef SA  #define SA  #include <iostream>  #include "Student.h"  using namespace std;  class StudentArray {  friend ostream& operator<<(ostream&, const StudentArray&);  public:  StudentArray(int size); // 생성자  StudentArray(const StudentArray& obj); // 복제 생성자  ~StudentArray(); // 소멸자  int size() const { return num\_students; }  Student& operator[] (int index) const;  void sortByBirthDate(); // 생일 정렬  void sortByName(); // 이름 정렬  void sortByST\_ID(); // 학번 정렬  void sortByGPA(); // 학점 정렬  private:  Student\* students;  int num\_students;  bool isValidIndex(int index) const; // 유효범위 확인  };  #endif // !SA |
| /\* StudentArray.cpp \*/  #include <iostream>  #include "StudentArray.h"  #include "Student.h"  using namespace std;  ostream& operator<<(ostream& fout, const StudentArray& sA) {  for (int i = 0; i < sA.num\_students; i++) {  fout << sA.students[i] << endl;  }  return fout;  }  StudentArray::StudentArray(int size) {  num\_students = size;  students = new Student[num\_students];  }  StudentArray::StudentArray(const StudentArray& obj) {  num\_students = obj.num\_students;  for (int i = 0; i < num\_students; i++)  students[i] = obj.students[i];  }  StudentArray::~StudentArray() {  /\*if (students != NULL)  delete[] students;\*/  }  Student& StudentArray::operator[] (int index) const {  if (isValidIndex(index))  return students[index];  else {  cout << "ERROR: Subscript out of range.₩n";  exit(0);  }  }  void StudentArray::sortByBirthDate() {  int i, j, mx;  Student tmp;  int min\_p;  Date min\_BD;  for (i = 0; i < num\_students - 1; i++) {  min\_p = i;  min\_BD = students[min\_p].birthDate;  for (j = i + 1; j < num\_students; j++) {  if (students[j].birthDate.getElapsedDays() < min\_BD.getElapsedDays()) {  min\_p = j;  min\_BD = students[j].birthDate;  }  }  if (min\_p != i) {  tmp = students[i];  students[i] = students[min\_p];  students[min\_p] = tmp;  }  }  }  void StudentArray::sortByName() {  int i, j, mx;  Student tmp;  int min\_p;  string min\_N;  for (i = 0; i < num\_students - 1; i++) {  min\_p = i;  min\_N = students[min\_p].name;  for (j = i + 1; j < num\_students; j++) {  if (students[j].name.compare(min\_N) < 0) {  min\_p = j;  min\_N = students[j].name;  }  }  if (min\_p != i) {  tmp = students[i];  students[i] = students[min\_p];  students[min\_p] = tmp;  }  }  }  void StudentArray::sortByST\_ID() {  int i, j, mx;  char minName[16] = { 0 };  Student tmp;  int min\_st, min\_ID;  for (i = 0; i < num\_students - 1; i++) {  min\_st = i;  min\_ID = students[min\_st].st\_id;  for (j = i + 1; j < num\_students; j++) {  if (students[j].st\_id < min\_ID) {  min\_st = j;  min\_ID = students[j].st\_id;  }  }  if (min\_st != i) {  tmp = students[i];  students[i] = students[min\_st];  students[min\_st] = tmp;  }  }  }  void StudentArray::sortByGPA() {  int i, j, mx;  char minName[16] = { 0 };  Student tmp;  int min\_st;  double min\_GPA;  for (i = 0; i < num\_students - 1; i++) {  min\_st = i;  min\_GPA = students[min\_st].st\_id;  for (j = i + 1; j < num\_students; j++) {  if (students[j].st\_id < min\_GPA) {  min\_st = j;  min\_GPA = students[j].gpa;  }  }  if (min\_st != i) {  tmp = students[i];  students[i] = students[min\_st];  students[min\_st] = tmp;  }  }  }  bool StudentArray::isValidIndex(int index) const {  if (index < 0 || index >= num\_students)  return false;  else  return true;  } |
|  |

**2. 2021-2 객체지향형 프로그래밍과 자료구조 실습 Oral Test**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 학번 | 21812158 | 성명 | 김주환 | 점수 |  |

|  |
| --- |
| **1. 객체 지향형 프로그래밍에서 상속 개념을 사용하는 장점은 무엇인가? 예를 들어 설명하라.**   * 소프트웨어 재사용(software-reuse)기능을 제공 상속 기반의 소프트웨어 개발에서는 자식 클래스는 이미 기능과 성능, 안정성이 검증되고 신뢰성이 높은 기존의 부모 클래스를 기반으로 새로운 클래스를 생성하기 때문에, 우수한 기능을 사용할 수 있다. 재사용을 통해서 개발 기간이나, 비용이 줄어든다.  다양하게 파생된 클래스를 사용하는 중에 공통적인 속성의 수정 또는 보완이 필요한 경우, 파생클래스가 상속받은 부모 클래스를 수정 및 보완함으로써 쉽게 정비할 수 있으며, 각 자식 클래스에서 공통적인 속성을 개별적으로 변경하는 것 보다 효율적이다.  회사의 경우, 먼저 일반화된 클래스를 설계한 후, 이를 기반으로 필요에 따라 특화된 클래스를 추가 생성함으로써 소프트웨어를 재사용할 수 있다. |
| **2. C++ 클래스에서 접근 지정자 “protected”와 “private”의 차이점에 대하여 설명하라.**   * 자식 클래스는 기반 클래스의 속성을 모두 상속받음 부모 클래스의 private member도 모두 상속받으나, 부모 클래스의 private member를 이름으로 직접 접근할 수는 없다. 부모 클래스의 private member는 부모 클래스의 public interface 멤버 함수를 통해서만 접근이 가능하다. * 상속관계에 있는 자식 클래스 구현에서 부모 클래스의 속성을 간편하게 접근하는 방법 “protected” 접근 지정자를 사용하여, 상속관계에 있는 자식 클래스 구현에서 부모 클래스의 속성을 간편하게 접근할 수 있도록 제한적으로 허용한다. * protected 접근 지정자 상속 관계에 있는 자식 클래스에 부모 클래스의 속성을 이름으로 직접 접근할 수 있도록 제한적으로 허용한다. 상속관계가 아닌 다른 클래스에서는 private처럼 관리되며, 이름으로 직접 접근할 수 없도록 보호된다. * protected 접근 지정자가 객체지향형 프로그래밍의 정보 보호 원칙을 위배? 자식 클래스에서 부모 클래스의 속성을 이름으로 직접 접근할 수 있도록 제한적으로 허용하는 것이 객체지향형 프로그래밍의 정보 보호 원칙을 위배한다는 의견도 있다. 구현에서의 간편성/편리성을 제공하기 위해서 자식 클래스에게만 제한적으로 허용한다. |
| **3. 상속받은 자식 클래스의 생성자 (constructor method of children class)에서 상속받는 부모 클래스의 생성자 (constructor method of parent class)를 호출하는 방법에 대하여 예를 들어 설명하라.**   * 상속된 자식 클래스의 복제 생성자에서 부모 클래스의 생성자를 호출 자식 클래스 객체에는 부모 클래스의 데이터 멤버를 함께 포함하고 있다. 자식 클래스의 초기화 색션에서 복제 대상 객체를 부모 클래스의 생성자에 전달하여 복제를 수행한다. * 파생 클래스의 생성자에서 인수로 그 클래스의 객체를 받는 경우, 파생 클래스의 생성자 초기화 색션에서 인수인 객체를 부모 클래스의 생성자에 전달하여 호출한다. * 생성자의 실행 순서 부모쪽의 생성자부터 실행하고 자식쪽으로 순서대로 실행하여 생성한다.  소멸자의 실행 순서는 생성자의 반대다.   ` |
| **4. 상속을 사용하는 C++ 클래스에서 상속이 되는 멤버함수와 상속이 되지 않는 멤버 함수들을 구분하여 각각 예를 들어 설명하라.**  부모 클래스의 멤버 함수들은 상속이 가능한데, 아래의 특별한 경우를 제외한 멤버함수들이 상속된다.   * 생성자 (constructors) & 소멸자 (destructors) 부모 클래스로부터 상속이 불가능하다. 함수 구현에서 초기화 색션에 추가하여 생성자를 호출 받아 사용 가능하다. * 복제 생성자 (copy constructor) 자식 클래스의 복제생성자 구현에서 부모 클래스의 복제생성자를 사용한다. 자식 클래스에서 따로 복제 생성자가 구현되지 않았다면 기본 복제함수가 사용된다. 하지만, 포인터나 동적 메모리 할당을 사용할 경우 클래스에 복제 생성자를 따로 구현해야한다. 생성자 중에서 전달되는 인수가 그 클래스의 객체인 경우 전달된 객체를 복사하여 새로운 객체를 생성한다. 인수나 반환이 객체거나 새 객체를 같은 클래스의 기존 객체와 똑같이 초기화할 경우 사용된다.      * 대입 연산자 (assignment operator, ‘=’) 자식 클래스의 대입연산자 구현에서 부모 클래스의 대입연산자를 사용한다. 자식 클래스에서 부모 클래스의 대입연산자가 필요한 경우, 자식 클래스의 대입연산자 구현에서 부모 클래스의 대입연산자를 구현해 주어야한다. |