|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_10 | 학번 : | 20135151 | 이름 : | 이갑성 |

* **강의 내용**

|  |
| --- |
| 1. 접근지정자에 따른 상속 : private |
| #include <iostream>  using namespace std;  class Base {  int a;  protected:  void setA(int a) { this->a = a; }  public:  void showA() { cout << a; }  };  class Derived : private Base {  int b;  protected:  void setB(int b) { this->b = b; }  public:  void showB() { cout << b; }  };  int main() {  Derived x;  x.a = 5; // ①  x.setA(10); // ②  x.showA(); // ③  x.b = 10; // ④  x.setB(10); // ⑤  x.showB(); // ⑥  } |
| **[실행결과]**  1번, 2번, 3번, 4번, 5번 private로 인하여 외부함수에서 접근 불가 |

|  |
| --- |
| 1. 접근지정자에 따른 상속 : protected |
| #include <iostream>  using namespace std;  class Base {  int a;  protected:  void setA(int a) { this->a = a; }  public:  void showA() { cout << a; }  };  class Derived : protected Base {  int b;  protected:  void setB(int b) { this->b = b; }  public:  void showB() { cout << b; }  };  int main() {  Derived x;  x.a = 5; // ①  x.setA(10); // ②  x.showA(); // ③  x.b = 10; // ④  x.setB(10); // ⑤  x.showB(); // ⑥  } |
| **[실행결과]**  1번, 2번, 3번, 4번, 5번 protected로 인하여 외부함수에서 접근 불가 |

|  |
| --- |
| 1. 함수 재정의 : //재정의일때는 자료형(class)에 따라 함수가 호출된다. |
| **class Base** {  public:  **void f()** { cout << "Base::f() called" << endl; }  };  **class Derived : public Base** {  public: **//함수 재정의**  **void f()** { cout << "Derived::f() called" << endl; }  };  void main() {  **Derived d**, \*pDer;  pDer = &d;  **pDer->f(); // Derived::f() 호출**  **//pDer->Base::f() …. Base::f() 호출**    Base\* pBase;  pBase = pDer; // 업캐스팅  **pBase->f(); // Base::f() 호출**  } |
| **[실행결과]** |

|  |
| --- |
| 1. 가상 함수와 오버라이딩 |
| **======= Circle.h =====**  #ifndef CIRCLE  #define CIRCLE  class Circle : public Shape {  protected:  virtual void draw();  };  #endif  **======= Line.h ====**  #ifndef LINE  #define LINE  class Line : public Shape {  protected:  virtual void draw();  };  #endif  **======== Rect.h =====**  #ifndef RECT  #define RECT  class Rect : public Shape {  protected:  virtual void draw();  };  #endif  **=======Shape.h ======**  #ifndef SHAPE  #define SHAPE  class Shape {  Shape\* next;  protected:  virtual void draw();  public:  Shape() { next = NULL;}  virtual ~Shape() { }  void paint();  Shape\* add(Shape\* p);  Shape\* getNext() { return next;}  };  #endif  **======== Circle.cpp =====**  #include <iostream>  #include "Shape.h"  #include "Circle.h"  using namespace std;  void Circle::draw() {  cout << "Circle" << endl;  }  **===== Line.cpp ====**  #include <iostream>  #include "Shape.h"  #include "Line.h"  using namespace std;  void Line::draw() {  cout << "Line" << endl;  }  =====Rect.cpp ====  #include <iostream>  #include "Shape.h"  #include "Rect.h"  using namespace std;  void Rect::draw() {  cout << "Rectangle" << endl;  }  **=== Shape.cpp ===**  #include <iostream>  #include "Shape.h"  using namespace std;    void Shape::paint() {  draw();  }  void Shape::draw() {  cout << "Shape" << endl;  }  Shape\* Shape::add(Shape \*p) {  this->next = p;  return p;  }  **==== main.cpp ====**  #include <iostream>  #include "Shape.h"  #include "Circle.h"  #include "Rect.h"  #include "Line.h"  using namespace std;  int main() {  Shape \*pStart=NULL;  Shape \*pLast;  pStart = new Circle(); // 처음에 원 도형을 생성한다.  pLast = pStart;  pLast = pLast->add(new Rect()); // 사각형 객체 생성  pLast = pLast->add(new Circle()); // 원 객체 생성  pLast = pLast->add(new Line()); // 선 객체 생성  pLast = pLast->add(new Rect()); // 사각형 객체 생성  // 현재 연결된 모든 도형을 화면에 그린다.  Shape\* p = pStart;  while(p != NULL) {  p->paint();  p = p->getNext();  }  // 현재 연결된 모든 도형을 삭제한다.  p = pStart;  while(p != NULL) {  Shape\* q = p->getNext(); // 다음 도형 주소 기억  delete p; // 기본 클래스의 가상 소멸자 호출  p = q; // 다음 도형 주소를 p에 저장  }  } |
| **[실행결과]** |

|  |
| --- |
| 1. 가상 함수와 오버라이딩 |
| #include <iostream>  using namespace std;  class Base {  public:  **virtual ~Base()** { cout << "~Base()" << endl; }  };  class Derived: public Base {  public:  **virtual ~Derived()** { cout << "~Derived()" << endl; }  };  int main() {  Derived \*dp = new Derived();  Base \*bp = new Derived();  **delete dp;** // Derived의 포인터로 소멸  **delete bp;** // Base의 포인터로 소멸  } |
| **[실행결과]** |

* 내용 점검

1. 교재 8장 연습문제

|  |  |
| --- | --- |
| 번호 | 문제풀이 |
| 1 | 2번 |
| 2 | 1번 |
| 3 | 3번 |
| 4 | 4번 |
| 5 | 2번 |
| 6 | 1번 2번 4번 |
| 7 | (1) up - 2 / down - 3  (2) p는 현재 부모 클래스의 주소를 가르키기 때문에 y를 접근할수없다. |
| 8 | (1) 4번  (2)  public:  int w;  int x;  int z;  (3) 3번  (4) dp = (D \*)ap; |
| 9 | (1) 생성자 A  생성자 B  (2) 생성자 A  생성자 B 10  (3) 생성자 A 32  생성자 B 400 |
| 10 | (1) 디폴트 생성자A가 없음  (2)  B() {  A(10);  cout << “생성자 B” << endl;  }  (3) B(int x) : A(x + 20) {  cout << “생성자 B ” << x << endl;  } |
| 11 | 3번 |
| 12 | 4번 |
| 13 | class Satellite : public Rocket, public Computer {  }; |
| 14 | (1)  class Hipen : public Pen, public Eraser  {  };  (2)  class OmniPen : public Pen, public Eraser, public Lock  {  }; |
| 15 | 4번 라인  죽음의 다이어몬드 때문에 중복으로 베이스 클래스가 생성되어 이중으로 power이 생성된다.  #include <iostream>  using namespace std;  class Vehicle {  public:  int power;  };  class Car : virtual public Vehicle {  public:  int color;  };  class Airplane : virtual public Vehicle{  public:  int altitude;  };  4  class FlyingCar : public Car, public Airplane {  public:  void go() {  cout << "난다요" << endl;  };  };  int main()  {  FlyingCar fCar;  fCar.go();  fCar.altitude = 2000;  fCar.color = 2000;  fCar.power - 2000;  } |
| 16 | #include <iostream>  using namespace std;  class TV {  public:  int screenSize;  };  class ColorTV : virtual public TV {  public:  int color;  };  class InternetTV : virtual public TV  {  public:  string ipAddr;  };  다중상속시 base클래스의 멤버변수가 중복 생성되기에 모호성이 발생되는 치명적은 단점이 발생된다.  이때 가상상속으로 인해 해결 모호성을 해결할 수 있다. |

* 프로그램 과제

|  |
| --- |
| 1. 교재 420p : 문제5번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class BaseArray  {  int capacity; //배열의 크기  int \*mem; //정수 배열을 만들기 위한 메모리의 포인터  protected:  BaseArray(int capacity = 100)  {  this->capacity = capacity;  mem = new int[capacity];  }  ~BaseArray() {  delete[] mem;  }  void put(int index, int val)  {  mem[index] = val;  }  int get(int index)  {  return mem[index];  }  int getCapacity()  {  return capacity;  }  };  class MyQueue : public BaseArray {  int size;  int index;  int popPoint;  public:  MyQueue(int size)  {  this->size = size;  index = 0;  popPoint = 0;  }  void enqueue(int n)  {  if (n > getCapacity())  {  cout << "용량을 초과하셨습니다. " << endl;  }  else {  put(index, n);  index++;  }  }  int capacity()  {  return getCapacity();  }  int length()  {  return index;  }  int dequeue()  {  int temp;  temp = get(popPoint);  popPoint++;  index--;  return temp;  }  };  int main()  {  MyQueue mQ(100);  int n;  cout << "큐에 삽입할 5개의 정수를 입력하라 >> ";  for (int i = 0; i < 5; i++)  {  cin >> n;  mQ.enqueue(n);  }  cout << "큐의 용량: " << mQ.capacity() << ", 큐의 크기: " << mQ.length() << endl;  cout << "큐의 원소를 순서대로 제거하여 출력한다 >> ";  while (mQ.length() != 0)  {  cout << mQ.dequeue() << ' ';  }  cout << endl << "큐의 현재 크기: " << mQ.length() << endl;  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 421p : 문제6번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class BaseArray  {  int capacity; //배열의 크기  int \*mem; //정수 배열을 만들기 위한 메모리의 포인터  protected:  BaseArray(int capacity = 100)  {  this->capacity = capacity;  mem = new int[capacity];  }  ~BaseArray() {  delete[] mem;  }  void put(int index, int val)  {  mem[index] = val;  }  int get(int index)  {  return mem[index];  }  int getCapacity()  {  return capacity;  }  };  class MyStack : public BaseArray  {  int size;  int index;  public:  MyStack(int size)  {  this->size = size;  index = 0;  }  void push(int n)  {  if (index > getCapacity())  {  cout << "용량을 초과하셨습니다" << endl;  }  else {  put(index, n);  index++;  }  }  int capacity()  {  return getCapacity();  }  int length()  {  return index;  }  int pop()  {  int temp;  index--;  temp = get(index);  return temp;  }  };  int main()  {  MyStack mStack(100);  int n;  cout << "스택에 삽입할 5개의 정수를 입력하라 >> ";  for (int i = 0; i < 5; i++)  {  cin >> n;  mStack.push(n);  }  cout << "스택의 용량: " << mStack.capacity() << " , 스택의 크기: " << mStack.length() << endl;  cout << "스택의 원소를 순서대로 제거하여 출력한다 >> ";  while (mStack.length() != 0)  {  cout << mStack.pop() << ' ';  }  cout << endl << "스택의 현재 크기 : " << mStack.length() << endl;  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 422p : 문제8번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Print  {  string model; //모델명  string manufacturer; //제조사  int nowPages; //용지  public:  Print(string model, string manufacturer, int pages)  {  this->model = model;  this->manufacturer = manufacturer;  this->nowPages = pages;  }    void printedCount(); //인쇄 매수를 정보  void availableCount(); //인쇄 종이 잔량를 정보  int print(int pages); //호출할때마다 pages매의 용지를 사용한다.    };  void Print::availableCount()  {  cout << nowPages << "장 ,";  }  void Print::printedCount()  {  cout << model << " ," + manufacturer << " ,남은 종이 ";  availableCount();  }  int Print::print(int pages)  {  int temp = nowPages;  nowPages = nowPages - pages;  if (nowPages < 0)  {  cout << "용지가 부족하여 프린트 하지 못합니다." << endl;  nowPages = temp;  return 0;  }  else  {  cout << "프린트하였습니다." << endl;  return 1;  }    }  class Ink : public Print {  int ink; //잉크  public:  Ink(string model, string manufacturer, int pages, int ink) : Print(model, manufacturer, pages)  {  this->ink = ink;  }  ~Ink() { }  void printedCount();  void availableInk(); // 잉크 잔량 정보  void printInkJet(int pages);  };  void Ink::availableInk()  {  cout << "남은 잉크 " << ink << endl;  }  void Ink::printedCount()  {  Print::printedCount();  availableInk();  }  void Ink::printInkJet(int pages)  {  if (print(pages))  {  this->ink = this->ink - pages;  }  }  class Laser : public Print {  int toner;  public:  Laser(string model, string manufacturer, int pages, int toner) : Print(model, manufacturer, pages)  {  this->toner = toner;  }  ~Laser() { }  void printedCount();  void availableToner(); //토너 잔량 정보  void printLaser(int pages);  };  void Laser::availableToner()  {  cout << "남은 토너 " << toner << endl;  }  void Laser::printedCount()  {  Print::printedCount();  availableToner();  }  void Laser::printLaser(int pages)  {  if (print(pages))  {  this->toner = this->toner - pages;  }  }  int main()  {  bool flag = true;  int menu, cnt;  char go;  Ink i1("Officejet V40", "HP", 5, 10); //잉크클래스의 객체 i1 생성  Laser l1("SCX-6x45", "삼성전자", 3, 20); //레이저클래스의 객체 l1 생성  cout << "현재 작동중인 2 대의 프린터는 아래와 같다." << endl;  cout << "잉크젯 : ";  i1.printedCount();  cout << "레이저 : ";  l1.printedCount();  while (flag)  {  cout << "프린터(1: 잉크젯, 2: 레이저)와 매수 입력 >> ";  cin >> menu >> cnt;  if (menu == 1)  {  i1.printInkJet(cnt);  }  else if (menu == 2)  {  l1.printLaser(cnt);  }  i1.printedCount();  l1.printedCount();  cout << "계속 프린트 하시겠습니까?(y/n) >> ";  cin >> go;  if (go == 'y')  flag = true;  else  flag = false;  }  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 423p 문제9번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Console {  public:  static int menu() {  int num;  cout << "\n예약:1, 취소:2, 보기:3, 끝내기: 4 >> ";  cin >> num;  return num;  }  static int time\_menu()  {  int num;  cout << "7시:1, 12시:2, 17시:3 >> ";  cin >> num;  return num;  }  };  class Seat {  string name;  int seatNum;  public:  Seat()  {  name = "---";  }  void setName(string name) //예약자 이름 저장  {  this->name = name;  }  string getName()  {  return name;  }  };  class Schedule {  Seat \*seat;  public:  Schedule()  {  seat = new Seat[8];  }  void showTime(int num) //시간의 예약자들 명단을 보여주는 함수  {  if (num == 1)  {  cout << "07시 : ";  }  else if (num == 2)  {  cout << "12시 : ";  }  else if (num == 3)  {  cout << "17시 : ";  }  for (int i = 0; i < 8; i++)  {  cout << seat[i].getName() << " ";  }  cout << endl;  }  string showReserveName(int num) //특정좌석의 예약자 이름을 보여주는 함수  {  return seat[num].getName();  }  void reserve(string name, int num) //예약하는 기능을 하는 함수  {  seat[num].setName(name);  }  };  class AirlineBook {  Schedule \*s;  public:  AirlineBook()  {  s = new Schedule[3];  }  void run()  {  bool flag = true;  int menu, flightNum, selectSeat;  string name;  while (flag)  {  menu = Console::menu();  switch (menu)  {  case 1:  flightNum = Console::time\_menu();  if (flightNum == 1 || flightNum == 2 || flightNum == 3)  {  s[flightNum - 1].showTime(flightNum);  cout << "좌석번호 >> ";  cin >> selectSeat;  cout << "이름 입력 >> ";  cin >> name;  s[flightNum - 1].reserve(name, selectSeat-1);  }  else  cout << "메뉴에 없는 번호를 입력하셨습니다. 다시 입력하세요";  break;  case 2:  flightNum = Console::time\_menu();  if (flightNum == 1 || flightNum == 2 || flightNum == 3)  {  s[flightNum - 1].showTime(flightNum);  cout << "좌석번호 >> ";  cin >> selectSeat;  cout << "이름 입력 >> ";  cin >> name;  s[flightNum - 1].reserve("---", selectSeat-1);  }  break;  case 3:  for (int i = 0; i < 3; i++)  {  s[i].showTime(i+1);  }  break;  case 4:  flag = false;  cout << "예약시스템을 종료합니다." << endl;  break;  }  }  }  };  int main()  {  AirlineBook \*air = new AirlineBook();  cout << "\*\*\*\*\*\*\*\* 한성한공에 오신것을 환영합니다. \*\*\*\*\*\*\*\*" << endl;  air->run();  } |
| **[실행결과]** |