|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_10 | 학번 : | 20145165 | 이름 : | 정균모 |

* **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]**  class Derived : private Base  상속을 private로 받기 때문에 기본 클래스 base에 맴버를 private로 계승한다. 그래서 main에서 Derived를 통해 base 맴버를 접근할 수 없다.  // ④X.b는 private이기 때문에 접근이 안되며  // ⑤protected는 파생클래스나 선언된 클래스에서만 접근이 가능해서 main에서는 접근이 불가능하다. |

|  |
| --- |
| 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]**  class Derived : protected Base  상속을 protected로 받기 때문에 기본 클래스 base에 맴버를 protected로 계승한다. 그래서 main에서 Derived를 통해 base 맴버를 접근할 수 없다.  // ④X.b는 private이기 때문에 접근이 안되며  // ⑤protected는 파생클래스나 선언된 클래스에서만 접근이 가능해서 main에서는 접근이 불가능하다. |

|  |
| --- |
| 1. 함수 재정의 : |
| **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 | 2 |
| 2 | 1 |
| 3 | 3 |
| 4 | 4 |
| 5 | 2 |
| 6 | 1, 2, 4 |
| 7 | (1)업: 2, 다운:3  (2) q는 상속받은 a에 접근하여 x 값을 변경할 수 있지만 b에 y 값을 변경할 수 는 없다. |
| 8 | (1)3  (2)z, x, w  (3)3  (4)에 = (D\*)ap; |
| 9 | (1) 생성자 B  (2) 생성자 B 32  (3) 생성자 A 32 생성자 B 400 |
| 10 | (1)디폴트 생성자가 없다.  (2) A a(20);  cout << "생성자 B " << endl;  (3) A a(x+20);  cout << "생성자 B " << x << endl; |
| 11 | 3 |
| 12 | 4 |
| 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 | 4 오류 수정: fcar.Vehicle::power = 200;  코드에서 다중 상속을 받으며 car 안에 Vehicle안에 power을 접근하는게 모호하기 때문에 확실히 어디 있는 power라는걸 알려줘야 한다. |
| 16 | Tv 클래스를 ColorTv와 InternetTv가 둘 다 접근하기 때문에 모호성 문제를 일으키기 쉽다. 수정 방법은 class ColorTv : vitual public Tv {}; 이렇게 가상 상속을 선언해주는 것 이다. |

* 프로그램 과제

|  |
| --- |
| 1. 교재 420p : 문제5번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class BaseArray {  int capacity;  int \*mem;  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 enindex;  int deindex;  public:  MyQueue(int size) : BaseArray(size) { enindex = 0; deindex = -1; }  void enqueue(int n) {  put(enindex, n);  enindex++;  }  int capacity() { return getCapacity(); }  int length() { return enindex; }  int dequeue() {  enindex--;  deindex++; return get(deindex);  }  };  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;  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 421p : 문제6번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class BaseArray {  int capacity;  int \*mem;  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 top;  public:  MyStack(int size) : BaseArray(size) { top = 0; }  void push(int n) { put(top, n); top++; }  int capacity() { return getCapacity(); }  int length() { return top; }  int pop() { top--; return get(top); }  };  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 << "스택용량:" << mStack.capacity() << ", 스택크기:" << mStack.length() << endl;  cout << "스택의 모든 원소를 팝하여 출력한다>> ";  while (mStack.length() != 0) {  cout << mStack.pop() << ' ';  }  cout << endl << "스택의 현재 크기 : " << mStack.length() << endl;  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 422p : 문제8번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class printer {  string model;  string manufacturer;  int printedCount;  int availableCount;  public:  printer(string m, string ma, int a) {  model = m;  manufacturer = ma;  availableCount = a;  }  string getModel() { return model; }  string getManu() { return manufacturer; }  int getAvail() { return availableCount; }  bool print(int pages) {  printedCount = pages;  if (availableCount >= printedCount) {  availableCount -= printedCount;  return true;  }  else {  cout << "용지가 부족하여 프린트할 수 없습니다." << endl;  return false;  }  }  };  class Ink : public printer {  int availableInk;  public:  Ink(string m, string ma, int a, int aInk) : printer(m, ma, a) {  availableInk = aInk;  }  void printInkJet(int pages) {  if (print(pages)) {  if (availableInk >= pages) {  availableInk -= pages;  cout << "프린트하였습니다." << endl;  }  }  }  void show() {  cout << getModel() << " ," << getManu() << " ,남은 종이 " << getAvail() << "장 ,남은 잉크 " << availableInk << endl;  }  };  class Laser : public printer {  int availableToner;  public:  Laser(string m, string ma, int a, int aToner) : printer(m, ma, a) {  availableToner = aToner;  }  void printLaser(int pages) {  if (print(pages)) {  if (availableToner >= pages) {  availableToner -= pages;  cout << "프린트하였습니다." << endl;  }  }  }  void show() {  cout << getModel() << " ," << getManu() << " ,남은 종이 " << getAvail() << "장 ,남은토너 " << availableToner << endl;  }  };  int main() {  Ink\* IP = new Ink("Officejet V40", "HP", 5, 10);  Laser\* LP = new Laser("SCX-6x45", "삼성전자", 3, 20);  int n1, n2;  char a;  cout << "현재 작동중인 2 대의 프린터는 아래와 같다" << endl;  cout << "잉크젯 : "; IP->show();  cout << "레이저 : "; LP->show();  while (1) {  cout << endl << "프린터(1:잉크젯, 2:레이저)와 매수 입력>>";  cin >> n1 >> n2;  if (n1 == 1)  {  IP->printInkJet(n2);  IP->show();  LP->show();  }  if (n1 == 2)  {  LP->printLaser(n2);  IP->show();  LP->show();  }  cout << "계속 프린트 하시겠습니까(y/n)>>";  cin >> a;  if (a == 'n') break;  }  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 423p 문제9번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Console {  public:  int menu() {  int num;  cout << "\n예약:1, 취소:2, 보기:3, 끝내기:4 >>";  cin >> num;  return num;  }  int time\_menu() {  int num;  cout << "7시:1, 12시:2, 17시:3 >>";  cin >> num;  return num;  }  };  class Seat {  string name;  int number;  public:  Seat() { this->name = "---"; this->number = -1; }  void setName(string name) { this->name = name; }  string getName() { return name; }  };  class Schedule {  Seat \*seat;  public:  Schedule() {  seat = new Seat[8];  }  void show(int num) {  if (num == 1) cout << "7시:";  else if (num == 2) cout << "12시:";  else if (num == 3) cout << "17시:";  for (int i = 0; i < 8; i++)  cout << "\t" << seat[i].getName();  cout << endl;  }  void setN(int num, string name) { seat[num - 1].setName(name); }  string getN(int num) { return seat[num - 1].getName(); }  };  class AirlineBook {  Schedule \*s;  public:  AirlineBook() {  s = new Schedule[3];  }  void program() {  bool exit = true;  int num;  Console c;  int seat\_num;  string name;  while (exit) {  num = c.menu();  if (num == 1) {  num = c.time\_menu();  if (num == 1 || num == 2 || num == 3) {  s[num - 1].show(num);  cout << "좌석 번호 >>"; cin >> seat\_num;  cin.ignore(1);  cout << "이름 입력 >>"; getline(cin, name);  s[num - 1].setN(seat\_num, name);  }  else  cout << "숫자를 잘못 입력하셨습니다." << endl;  }  else if (num == 2) {  num = c.time\_menu();  if (num == 1 || num == 2 || num == 3) {  s[num - 1].show(num);  cout << "좌석 번호 >>"; cin >> seat\_num;  cin.ignore(1);  cout << "이름 입력 >>"; getline(cin, name);  if (name != s[num - 1].getN(seat\_num))  cout << "해당 좌석에 예약되어 있는 이름과 일치하지 않습니다." << endl;  else  s[num - 1].setN(seat\_num, "---");  }  else  cout << "숫자를 잘못 입력하셨습니다." << endl;  }  else if (num == 3) {  s[0].show(1);  s[1].show(2);  s[2].show(3);  }  else if (num == 4) {  cout << "예약 시스템을 종료합니다." << endl;  exit = false;  }  else  cout << "숫자를 잘못 입력하셨습니다." << endl;  }  }  };  int main() {  AirlineBook \*Air = new AirlineBook();  cout << "\*\*\* 한성한공에 오신것을 환영합니다. \*\*\*\n";  Air->program();  system("pause");  } |
| **[실행결과]** |