Ch01. C

1. standard output

#include <iostream>

using namespace std;

int main ( ) {

cout << "Welcome to Object-Oriented" << endl;

}

1. Function Overloading

함수호출시 ‘함수의 이름’과 ‘전달되는 인자의 정보(매개변수의 자료형과 수)’를 동시에 참조하여 호출할 함수를 결정한다. 따라서 이렇듯 매개변수의 선언이 다르다면 동일한 이름의 함수도 정의가능하다.

1. 매개변수 디폴트 값

함수의 선언부에만 표시하도록 하고, 일부에만 지정할 경우 인자가 왼쪽에서부터 채워지므로, 디폴트는 오른쪽부터 채워져야 한다.

void display(int, int = 5, int = 0);

1. 인라인 함수 \*
2. Namespaces

namespace sdds {

}

using namespace sdds;

Ch02. C2

1. Bool

The bool type stores a logical value: true or false.

The ! operator reverses that value: !true is false and !false is true.

! is self-inverting on bool types, but not self-inverting on other types.

int x = 4;

cout << !!x; === 1 (true) / 0 (false)

1. Reference
2. 참조자와 함수
3. Dynamic Allocation

b\_flowers = new Flower [flowerCount];

SimpleClass \* ptr = new SimpleClass(30); // 생성자에 30 전달

Dynamic Deallocation

delete[] b\_flowers;

1. 표준함수 호출

Ch03. Class

1. Structure
2. Class & Object
3. OOP

Ch04. Class 2

1. Information hiding

클래스의멤버변수를private으로선언하고, 해당변수에접근하는함수를별도로정의해서, 안젂한형태로멤버변수의접근을유도하는것이바로‘정보은닉’이며, 이는좋은클래스가되기위한기본조건이된다!

const 함수내에서는동일클래스에선언된멤버변수의값을변경하지못한다!

1. Encapsulation 관련있는모든것을하나의클래스안에묶어두는것!
2. Constructor & Destructor

Constructor

클래스의이름과동일한이름의함수이면서반환형이선언되지않았고실제로반환하지않는함수를가리켜생성자라한다! – 초기화에 사용 (오버로딩 가능하고 디폴트 설정이 가능)

* 멤버이니셜라이저는함수의선언부가아닌, 정의부에명시한다.
* 이니셜라이저의초기화는선언과동시에초기화되는형태이므로, 참조자의초기화도가능하다!
* 정의하지 않으면 자동으로 생성 / 하나라도 정의되면 디폴트 생성자는 추가되지 않는다. 이 경우 인자를 받지 않는 생성자 호출이 불가하다.
* AAA 클래스의멤버함수내에서도AAA 클래스의객체생성이가능하다!

Book::Book() : PubRecord(), m\_ISBN(0), m\_memberID(0) {};

Book::Book(const Book& book) :PubRecord(book), m\_ISBN(book.m\_ISBN), m\_memberID(0), m\_date(book.m\_date) {}

Destructor

생성자와마찬가지로소멸자도정의하지않으면디폴트소멸자가삽입된다. 동적할당의 경우 메모리 공간 소멸시키는 좋은 위치가 소멸자이다.

1. Class & Array & This pointer

객체배열! 객체로이뤄진배열, 따라서배열생성시객체가함께생성된다.

이경우호출되는생성자는void 생성자

객체포인터배열! 객체를저장할수있는포인터변수로이뤄진배열! 따라서별도의객체생성과정을거쳐야한다.

* this 포인터는그값이결정되어있지않은포인터이다. 왜냐하면this 포인터는this가사용된객체자신의주소값을정보로담고있는포인터이기때문이다.

Student Student::display() const {

// ...

return \*this;

}

const Student& Student::display() const {

// ...

return \*this;

}

Ch05. Copy Constructor

1. Copy Constructor / 정의 없으면 자동 삽입된다.

Student(const Student& copy) : num1(copy.num1), num2(copy.num2){}

SoSimple sim2=sim1 은묵시적으로SoSimple sim2(sim1) 으로해석이된다.

Student::Student(const Student& src) {

// shallow copies

no = src.no;

ng = src.ng;

// allocate dynamic memory for grades

if (src.grade != nullptr) {

grade = new float[ng];

// copy data from the source resource

// to the newly allocated resource

for (int i = 0; i < ng; i++)

grade[i] = src.grade[i];

}

else {

grade = nullptr;

}

}

1. shallow copies & Deep Copy & Copy Assignment Operator

Copy Assignment Operator

Student& Student::operator=(const Student& source) {

// check for self-assignment

if (this != &source) {

// shallow copy non-resource variables

no = source.no;

ng = source.ng;

// deallocate previously allocated dynamic memory

delete [] grade;

// allocate new dynamic memory, if needed

if (source.grade != nullptr) {

grade = new float[ng];

// copy the resource data

for (int i = 0; i < ng; i++)

grade[i] = source.grade[i];

}

else {

grade = nullptr;

}

}

return \*this;

}

복사금지

Student(const Student& source) = delete;

Student& operator=(const Student& source) = delete;

1. 호출시점







Ch06. Friend & Static & Const

\*\*const로선언된객체를대상으로는const로선언되지않는멤버함수의호출이불가능하다.

friend 선언은private 멤버의접근을허용하는선언이다.

class MenuItem {

friend class Menu; // 메뉴 클래스에서 메뉴아이템의 멤버에 직접접근 가능하다

char\* m\_menuItemStr;

MenuItem();

~MenuItem();

MenuItem(const char\* menuItemStr);

MenuItem(const MenuItem&) = delete;

MenuItem& operator=(const MenuItem&) = delete;

operator bool() const;

operator const char\* () const;

std::ostream& display(std::ostream& os = std::cout) const;

};

클래스 밖의 함수도 friend 선언이 가능하다.

Ch07. Inheritance \*



이니셜라이저를통해서유도클래스는기초클래스의생성자를명시적으로호출해야한다.

유도클래스의생성자는기초클래스의멤버를초기화할의무를갖는다. 단! 기초클래스의생성자를명시적으로호출해서초기화해야한다.





IS - A관계 / HAS – A 관계는 포함으로 표현

Ch08. Polymorphism

1. 객체포인터의 참조관계

 

IS - A관계 = 학생은 사람이다.

Person \* ptr=new Student();

Person \* ptr=new PartTimeStudent();

Student \* ptr=new PartTimeStudent();

“C++에서, AAA형포인터변수는AAA 객체또는AAA를직접혹은간접적으로상속하는모든객체를가리킬수있다(객체의주소값을저장할수있다).”

1. 가상함수

“C++ 컴파일러는포인터를이용한연산의가능성여부를판단할때, 포인터의자료형을기준으로판단하지, 실제가리키는객체의자료형을기준으로판단하지않는다.” 따라서포인터형에해당하는클래스의멤버에만접근이가능하다.

하지만, virtual 이라는 문자를 함수 앞에 붙이면, 포인터의형에상관없이포인터가가리키는객체의마지막오버라이딩함수를호출한다.



몸체가정의되지않은함수를가리켜순수가상함수라하며, Pure Virtual Function

하나이상의순수가상함수를멤버로두어서객체생성이불가능한클래스를가리켜추상클래스라한다. Abstract Classes



virtual void display(std::ostream&) const = 0;

#include <iostream>

#include "iPerson.h"

const int NC = 30;

const int NG = 20;

class Person : public iPerson {

char name[NC+1];

public:

Person();

Person(const char\*);

void display(std::ostream&) const;

};

class Student : public Person {

int no;

float grade[NG];

int ng;

public:

Student();

Student(int);

Student(const char\*, int, const float\*, int);

void display(std::ostream&) const;

};

1. 가상 소멸자와 참조자의 참조가능성

소멸자에 항상 virtual 붙이기

Ch09. Virtual Functions

1. 멤버함수와 가상함수의 동작원리
2. 다중상속에 대한 이해

Ch10. Operator Overloading

Ch11. Operator Overloading2

Ch12. String Class

Ch13. Template1

Ch14. Template2

Ch15. Exception Handling

Ch16. 형변환 연산자 \*