

1. 용어 및 함수 정리

<책 정리>

- **Up/Down Casting** : 상속 관계에 있는 클래스 사이의 형변 환은 업캐스팅, 다운캐스팅 2가지로 구분된다.

- **UpCasting(업캐스팅)** : 기반 클래스의 포인터 변수가 파생 클래스의 인스턴스를 가리킬 수 있는데, 이를 업 캐스팅이라 하며 컴파일러에서 자동으로 형변환이 이루어진다. 파생 객체의 포인터가 기반 객체의 포인터로 형변환 하는 것이 다. 참조 가능한 영역이 축소된다. 컴파일러에 의해서 자동으로 형변환된다.

- **DownCasting(다운캐스팅)** : 파생 클래스로 선언된 포인터 변수에 기반 클래스로 선언된 객체의 주소를 저장하는 것 파생 클래스로 형변환 하는 것이다.

* 참조 가능한 영역이 확대되는 것을 의미한다.
* 컴파일러에 의해서 자동으로 형변환되지 않는다.
* 프로그래머에 의해서 명시적으로 캐스팅을 해 주어야만 한다.

* 형변환 후에도 실행시 예외사항이 발생할 수 있으므로 인스턴스 의 클래스형과 참조하는 포인터 변수의 상속 관계를 생각해서 명시적 캐스팅을 해야 한다.

- **Template Function(템플릿 함수)** : 코드의 재활용과 다형성 을 보장하는 함수이다. 함수의 오버로딩으로 함수의 매개변수에 따라 여러 함수를 정의해야하지만 템플릿 함수를 이용하면 한번만 정의하여 사용가능하다.

- **가상 기본 클래스** : 클래스를 두 번 상속받아도 한 번만 상속 하도록 해주는 클래스

- **가상 기본 클래스가 필요한 이유** : 다중 상속 중에 나타날 수 있는 모호성의 문제를 해결하기 위해서

- **묵시적 캐스팅** : 컴파일러에 의해서 자동으로 형 변환이 일어 나는 것을 말한다. - 다중 상속 : 여러 개의 기본 클래스에서 기능을 상속하여 파 생 클래스를 작성하는 것

- **Exception Handling(예외 처리)** : 수행 도중 발생하는 예외 상황을 처리할 수 있도록 하는 기능

- **Binding(바인딩)** : 함수 호출을 해당 함수의 정의와 결합해 둔 것이다.

- **Static Binding(정적 바인딩)** : Early Binding(이른 바인딩)이 라고도 하며 컴파일 할 때 미리 호출될 함수가 결정되는 것

- **Dynamic Binding(동적 바인딩)** : Lately Binding(늦은 바인딩)이라고도 하며 실행할 때 호출할 멤버함수를 결정

해서 정확 하고 이치에 맞는 멤버함수를 호출하도록 한다.

- **Virtual Function(가상함수)** : virtual 예약어를 붙인 함수를 가상함수라고 한다. 클래스 내의 멤버함수일 경우에 만 지정할 수 있으며 가상함수의 특징은 상속된다.

- **가상함수가 필요한 이유** : 멤버함수를 호출할 때 포인터 변수 의 자료형에 의존하지 않게 된다.

- **가상함수의 장점** : 가상함수는 융통성 있게 함수를 호출한다 는 장점이 있다.

- **가상함수의 단점** : 가상함수를 사용하려면 가상 테이블을 유 지하려고 가상함수만큼 주소 배열을 만들어 주므로 메모리에 부 담을 줄 수 있다. 각 객체도 가상함수 테이블의 주소를 저장하는 가상 포인터 변 수를 위한 메모리를 할당해야 한다. 가상함수가 호출될 때마다 가상함수 테이블을 이용해서 주소를 조사해야 하므로 처리속도가 지연된다.

- **Pure Virtual Function(완전 가상함수)** : 함수의 정의 없이 함수의 유형만을 기반 클래스에 제시해 놓는 것이다. 가상함수처럼 멤버함수를 선언할 때 예약어 virtual을 선언문의 맨 앞에 붙이고 함수의 선언문 마지막 부분에 '=0'을 덧붙인다.

- **Abstract Class(추상 클래스)** : 완전 가상함수를 하나 이상 포함한 클래스로, 객체를 생성할 수 없으며 상속을 위한 기반 클래스로 사용된다.

- **Function Overloading(함수 오버로딩)** : 매개변수의 자료형 또는 개수가 다르면 동일한 이름의 함수정의를 허용

- **Operator Overloading(연산자 오버로딩)** : 이미 사용 중인 연산자를 다른 용도로 다시 정의해서 사용할 수 있다.

- **Protected** : 상속 가능한 클래스 내부에서는 public이고 외부 클래스에서는 private처럼 사용된다. 기반 클래스의 멤버를 protected 접근 지정자로 선언하면 그 멤버를 파생 클래스에서는 아무런 조건 없이 사용하고 다른 클래스(클래스 외부)에서는 사용할 수 없게 된다.

<피피티 정리>

- **입력 스트림의 버퍼** : 입력장치로부터 입력된 데이터를 프로그램으로 전달하기 전에 일시 저장

- **출력 스트림의 버퍼** : 프로그램에서 출력된 데이터를 출력 장치로 보내기 전에 일시 저장

- **cin** : istream타입의 스트림 객체로서 키보드 장치와 연

결

- **cout** : ostream타입의 스트림 객체로서 스크린 장치와 연결

- **cerr** : ostream타입의 스트림 객체로서 스크린 장치와 연결, 오류 메시지를 출력할 목적, 스트림 내부 버퍼 거치지 않고 출력

- **clog** : ostream타입의 스트림 객체로서 스크린 장치와 연결, 오류 메시지를 출력할 목적, 스트림 내부에 버퍼 거쳐 출력

- **ostream** 멤버 함수 이용한 문자 출력 :

- 1) cout.put(33) -> '!'출력(문자하나 출력)
- 2) cout.write(str, 6) -> str에 있는 6개 문자 출력
- 3) cout.flush() -> 버퍼 내용 강제 출력

- **istream** 이용한 입력

- 1) cin.get(ch)
- 2) ch=cin.get()

- **cin.ignore()/cin.get()** : 스트림 버퍼에 남은거 읽어서 버리는 역할

- **getline()**으로 한 줄 단위로 문장 읽기 : cin.get()과 달리 'Wn'은 line에 삽입하지 않고, 스트림 버퍼에서 제거 enter키가 남지 않는다.

-**cin.ignore(10)** : 입력스트림에 입력된 문자중 10개 제거

-**int n = cin.gcount()** : 최근의 실행한 함수에서 읽은 문자의 개수 리턴

- **포맷 플래그** : 입출력 스트림에서 입출력 형식을 지정하기 위한 플래그

플래그	값	의미
ios::skipws	0x0001	입력시 공백 문자(스페이스, 탭, 개행문자)를 무시
ios::unitbuf	0x0002	출력 스트림에 들어오는 데이터를 버퍼링하지 않고 바로 출력
ios::uppercase	0x0004	16진수의 A~F, 지수 표현의 E를 대문자로 출력
ios::showbase	0x0008	16진수이면 0x를, 8진수이면 0을 숫자 앞에 붙여 출력
ios::showpoint	0x0010	실수 값에 대해, 정수 부분과 더불어 소수점 이하의 끝자리들을 0으로 출력
ios::showpos	0x0020	양수에 대해 + 기호 출력
ios::left	0x0040	필드를 왼쪽 맞춤(left-align) 형식으로 출력
ios::right	0x0080	필드를 오른쪽 맞춤(right-align) 형식으로 출력
ios::internal	0x0100	부호는 왼쪽 맞춤으로 숫자는 오른쪽 맞춤으로 출력
ios::dec	0x0200	10진수로 출력, 디폴트 설정
ios::oct	0x0400	8진수로 출력
ios::hex	0x0800	16진수로 출력
ios::scientific	0x1000	실수에 대해 과학 산술용 규칙에 따라 출력
ios::fixed	0x2000	실수에 대해 소수점 형태로 출력
ios::boolalpha	0x4000	설정되면, 논리값 true를 "true"로, false를 "false"로 출력하고, 설정되지 않으면, 정수 1과 0으로 출력

- **ios** 클래스에 지정된 포맷 플래그

- **unsetf** : 포맷플래그 해제하고 이전 플래그 리턴

ex) cout.unsetf(ios::dec) : 10진수 해제

- **setf** : 포맷플래그 설정하고 이전 플래그 리턴

ex) cout.setf(ios::hex) : 16진수로 설정

- **조작자(스트림 조작자)** : 입출력 포맷 지정 목적으로 조작자는 함수이다.

* 매개 변수 없는 조작자 :

cout << hex << showbase << 30 << endl; - 0x1e

* 매개 변수 있는 조작자

⊙ #include <iomanip> 필요

cout << setw(10) << setfill('^') << "Hello" << endl;
-> ^^^^^Hello

-매개 변수를 가진 조작자

조작자	I/O	용도
resetioflags(long flags)	IO	flags에 지정된 플래그들 해제
setbase(int base)	0	base를 출력할 수의 진수로 지정
setfill(char cFill)	I	필드를 출력하고 남은 공간에 cFill 문자로 채움
setioflags(long flags)	IO	flags를 스트림 입출력 플래그로 설정
setprecision(int np)	0	출력되는 수의 유효 숫자 자리수를 np개로 설정, 소수점(.)은 별도로 카운트
setw(int minWidth)	0	필드의 최소 너비를 minWidth로 지정

<서술형 부분>

#1. 조작자 사용하지 않은 경우

```
#include<iostream>
#include<iomanip>
using namespace std;

void main()
{
    int i,j,k;
    for(i=0;i<26;i++)
    {
        for(j=65;j<=65+i;j++)
        {
            cout.put(j);
        }
        cout.put('\n');
    }
}
```

```
A
AB
ABC
ABCD
ABCDE
ABCDEF
ABCDEFG
ABCDEFGH
ABCDEFGH I
ABCDEFGH I J
ABCDEFGH I JK
ABCDEFGH I JKL
ABCDEFGH I JKLM
ABCDEFGH I JKLMN
ABCDEFGH I JKLMNO
ABCDEFGH I JKLMNOP
ABCDEFGH I JKLMNOPQ
ABCDEFGH I JKLMNOPQR
ABCDEFGH I JKLMNOPQRS
ABCDEFGH I JKLMNOPQRST
ABCDEFGH I JKLMNOPQRSTU
ABCDEFGH I JKLMNOPQRSTU V
ABCDEFGH I JKLMNOPQRSTU VW
ABCDEFGH I JKLMNOPQRSTU VWX
ABCDEFGH I JKLMNOPQRSTU VWXY
ABCDEFGH I JKLMNOPQRSTU VWXYZ
계속하려면 아무 키나 누르십시오 . . .
```

#1-1

```
#include<iostream>
#include<iomanip>
using namespace std;

void main()
{
    int i,j,k;
    for(i=0;i<26;i++)
    {
        for(j=90;j>=90-i;j--)
        {
            cout.put(j);
        }
        cout.put('\n');
    }
}
```

```
Z
ZY
ZYX
ZYXW
ZYXWV
ZYXWVU
ZYXWVUT
ZYXWVUTS
ZYXWVUTSR
ZYXWVUTSRQ
ZYXWVUTSRQP
ZYXWVUTSRQP O
ZYXWVUTSRQP ON
ZYXWVUTSRQP ONM
ZYXWVUTSRQP ONML
ZYXWVUTSRQP ONMLK
ZYXWVUTSRQP ONMLKJ
ZYXWVUTSRQP ONMLKJ I
ZYXWVUTSRQP ONMLKJ IH
ZYXWVUTSRQP ONMLKJ IHG
ZYXWVUTSRQP ONMLKJ IHGF
ZYXWVUTSRQP ONMLKJ IHGF E
ZYXWVUTSRQP ONMLKJ IHGFEDC
ZYXWVUTSRQP ONMLKJ IHGFEDCB
ZYXWVUTSRQP ONMLKJ IHGFEDCBA
계속하려면 아무 키나 누르십시오 . . .
```

#1-2

```
#include<iostream>
#include<iomanip>
using namespace std;

void main()
{
    int i,j,k;
    for(i=0;i<26;i++)
    {
        for(j=65;j<=90-i;j++)
        {
            cout.put(j);
        }
        cout.put('\n');
    }
}
```

```
ABCDEFGHIJKLMNPOQRSTUVWXYZ
ABCDEFGHIJKLMNPOQRSTUVWXYZ
ABCDEFGHIJKLMNPOQRSTUVW
ABCDEFGHIJKLMNPOQRSTUVW
ABCDEFGHIJKLMNPOQRSTUV
ABCDEFGHIJKLMNPOQRSTU
ABCDEFGHIJKLMNPOQRST
ABCDEFGHIJKLMNPOQRS
ABCDEFGHIJKLMNPOQR
ABCDEFGHIJKLMNOPQ
ABCDEFGHIJKLMNOP
ABCDEFGHIJKLMNO
ABCDEFGHIJKLMN
ABCDEFGHIJKLM
ABCDEFGHIJKL
ABCDEFGHIJK
ABCDEFGHIJ
ABCDEFGHI
ABCDEFGH
ABCDEFG
ABCDEF
ABCDE
ABCD
ABC
AB
A
계속하려면 아무 키나 누르십시오 . . .
```

#1-3

```
#include<iostream>
#include<iomanip>
using namespace std;

void main()
{
    int i,j,k;
    for(i=0;i<26;i++)
    {
        for(j=65;j<90-i;j++)
        {
            cout.put(' ');
        }
        cout.put(65+i);
        cout.put('\n');
    }
}
```

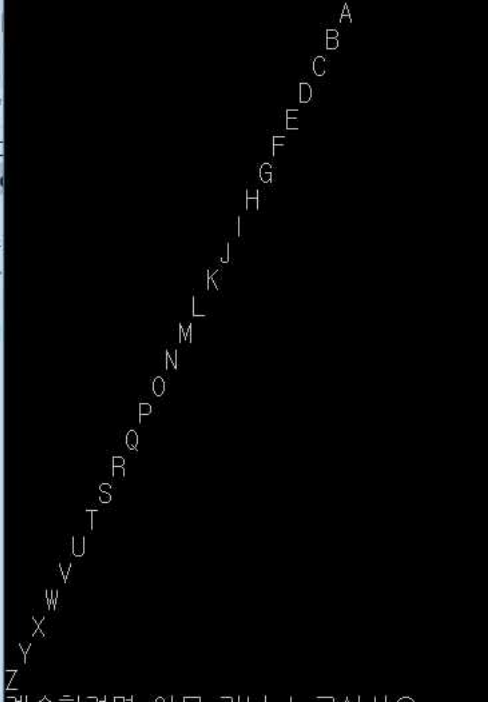
```

                                     A
                                    B
                                   C
                                  D
                                 E
                                F
                               G
                              H
                             I
                            J
                           K
                          L
                         M
                        N
                       O
                      P
                     Q
                    R
                   S
                  T
                 U
                V
               W
              X
             Y
            Z
계속하려면 아무 키나 누르십시오 . . .
```

#2. 조작자 사용하는 경우

```
#include<iostream>
#include<iomanip>
using namespace std;

void main()
{
    int i,j;
    for(i=0;i<26;i++)
    {
        cout<<setw(26-i)<<setfill(' ')<<char(65+i);
        cout.put('\n');
    }
}
```



```
A
 B
  C
   D
    E
     F
      G
       H
        I
         J
          K
           L
            M
             N
              O
               P
                Q
                 R
                  S
                   T
                    U
                     V
                      W
                       X
                        Y
                         Z
```