Lab 1

Basic of C++

input & output for C++

```cpp
#include <iostream>
#include <string>

int main(){
    std::string a;
    std::getline(std::cin, a);
    std::cout << a << std::endl;
    int b;
    std::cin>>b;
    std::cout << b <<std::endl;
    return 0;
}

```

- string 을 읽어오는 부분에서 cin으로 하면서 “hello world"를 치면 결과가 어떻게 나오는지도 확인해보세요.

```
PS C:\vscodepractice\test_C++\helloworldcpp\cin> g++ cin.cpp -o cin
PS C:\vscodepractice\test_C++\helloworldcpp\cin> ./cin
hello world
hello world
2
2
PS C:\vscodepractice\test_C++\helloworldcpp\cin>
```

Basic of C++

input & output for C++

```cpp
#include <iostream>
#include <string>
using namespace std;

int main(){
    string a;
    getline(cin, a);
    cout << a << endl;
    int b;
    cin>>b;
    cout << b << endl;
    return 0;
}

```

- string 을 읽어오는 부분에서 cin으로 하면서 “hello world"를 치면 결과가 어떻게 나오는지도 확인해보세요.

```
PS C:\vscodepractice\test_C++\helloworldcpp\cin> g++ cin.cpp
 -o cin
PS C:\vscodepractice\test_C++\helloworldcpp\cin> .\cin.exe
hello world
hello world
hello world
0
PS C:\vscodepractice\test_C++\helloworldcpp\cin>
```

Function Overloading

```
#include <iostream>

void function(void)
{
        std::cout << "function(void) call" << std::endl;
}

void function(int a, int b)
{
        std::cout << "function("<<a<<","<<b<<") call"<<std::endl;
}

int main(void)
{
        function();
        function(12, 13);
        return 0;
}
```

- 같은 함수이름 다른 매개변수 값을 작성

```
PS C:\vscodepractice\test_C++\helloworldcpp\cin> .\foverloading.exe
function(void) call
function(12,13) call
PS C:\vscodepractice\test_C++\helloworldcpp\cin>
```

! 같은 이름의 함수는 C언어에서는 오류-> C++에서는 가능함: 함수이름 뿐만 아니라 매개변수의 정보까지도 동시에 참조함

Default Parameter

```
#include <iostream>

using namespace std;


int func(int a = 0)
{
        return a+1;
}

int main(void)
{
        cout << func(11) << endl;
        cout << func() << endl;
        return 0;
}
```

a값에 매개변수가 없으면 a=0으로 자동할당

- 기본 매개변수가 설정되어 있으면 함수 호출 시 실제 매개변수를 넣지 않아도 해당 매개변수의 값이 자동으로 기본값으로 할당됩니다.

```
2  using namespace std;
3  int func(int a = 2)
4  {
5      cout << a << endl;
6      return a+1;
7  }
8
9  int main(void)
10 {
11     cout << func(11) << endl;
12     cout << func() << endl;  return 0;
13 }
14
```

값 a=11

```
PS C:\vscodepractice\test_C++\helloworldcpp\cin> ./fo
11
12
2
3
PS C:\vscodepractice\test_C++\helloworldcpp\cin>
```

```
PS C:\vscodepractice\test_C++\helloworldcpp\cin> g++ .\functionoverloading.cpp -o fo
PS C:\vscodepractice\test_C++\helloworldcpp\cin> ./fo
12
1
PS C:\vscodepractice\test_C++\helloworldcpp\cin>
```

C++ Basics & Data Types

signed: 부호 있는 정수를 표현합니다. 보통 signed 키워드는 생략합니다.
unsigned: 부호 없는 정수를 표현합니다. 따라서 값은 0부터 시작하게 됩니다.

	자료형	크기	범위
실수형	long double	8byte (64bit)	±1.7×10^(-307) ~ ±1.7×10^(308) 이상
	double	8byte (64bit)	±1.7×10^(-307) ~ ±1.7×10^(308)
	float	4byte (32bit)	±3.4×10^(-37) ~ ±3.4×10^(38)
정수형	unsigned long (int)	8 byte (64 bit)	0 ~ 18,446,744,073,709,551,615
	(signed) long (int)	8 byte (64 bit)	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
	unsigned int	4 byte (32 bit)	0 ~ 4,294,967,295
	(signed) int	4 byte (32 bit)	-2,147,483,648 ~ 2,147,483,647
	unsigned short (int)	2 byte (16 bit)	0 ~ 65,535
	(signed) short (int)	2 byte (16 bit)	-32,768 ~ 32,767
문자형	unsigned char	1 byte (8 bit)	0 ~ 255
	(signed) char	1 byte (8 bit)	-128 ~ 127
논리형	bool	1 byte (8 bit)	0 ~ 1

C++ Basics & Data Types

- 문자형 char
  - char A = 'A';
- 정수형 int
  - int A = 10;
- 실수형 float
  - float A = 12.34;
- bool형 : true / false 가리키는 데이터형
  - bool A = 0;
  - bool A = false;

C++ Basics & Data Types

- 문자형 string : 문자의 모음
  String: collection of chars
  #include <string> 해줘야함.
  - string A = “hello”;

- 포인터 *&:
  데이터의 주소를 저장하는 변수
  int B;
  - int *A; // 포인터 변수생성
  - A = &B; //포인터 변수 A에B의 주소 저장
  - *A = 100; (*A의 주소에 값100 을저장)

- 배열 : 같은 타입의 데이터의 집합
  array: collection of data of same type
  - int A[200] = {1, 2, 3};

```
int main(){
    int B;
    int *A; //포인터 변수
    A = &B; //포인터 변수 A에 B의 주소를 저장
    cin>>B;
    cout<<"adress "<<*A<<endl;
    cout<<"number: "<<A<<endl;
    cin>>B;
    cout<<"adress "<<*A<<endl;
    cout<<"number: "<<A<<endl;
    return 0;
}
```

```
10
adress 10
number: 0x61ff08
20
adress 20
number: 0x61ff08
```

Exercise 2 - 1

- 2개의 정수 값과 2개의 char값을 넣고 swap된 결과를 프린트해보기

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

void swap(int *a, int *b);
void swap   (char *a, char *b);


int main(void)
{
// FILL IN
}
void swap(int *a, int *b)
{
// FILL IN

}
void swap(char *a, char *b)
{
//FILL IN
}
```

```
put two numbers for swapping
1 2
put two characters for swapping
a b
number swap!!!
character swap!!
result
2 1
b a
```

swap 함수 내부에서 출력.

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

void swap(int *a, int *b);  void swap(char *a, char *b);
int main(void)
{
// FILL IN

    char x,y;

    int c1, c2;
    std::cout<<"put two numbers for swapping"<<endl;
    std::cin>>c1>>c2;
    std::cout<<"put two characters for swapping"<<endl;
    std::cin>>x>>y;
    swap(c1,c2);
    swap(x,y);
    std::cout<<"result"<<endl;
    std::cout<<c1<<" ";
    std::cout<<c2<<endl;
    std::cout<<x<<" ";
    std::cout<<y<<endl;


}
```

```
void swap(int *a, int *b)
{
// FILL IN
    int temp=*a;
    *a=*b;
    *b=temp;
    cout<<"number swap!!"<<endl;
}
void swap(char *a, char *b)
{
//FILL IN
    char temp=*a;
    *a=*b;
    *b=temp;
    cout<<"character swap!!"<<endl;
}
```

Exercise 2 - 2

- Character값을 받고 소문자를 받으면 대문자로 변환, 대문자를 받으면 소문자로 변환해서 변환된 값을 출력 하기
- HINT : ASCII code : A = 65 / a= 97

```
#include <iostream>

using namespace std;

void convertAlphabet(char alphabet);
{
        //Fill in
}
int main(void)
{
        char alphabet;
        std::cout<<"Enter Capital or Small letter: "<<endl;
        std::cin>>alphabet;
        convertAlphabet(alphabet);
        return 0;
}
```

```
PS C:\vscodepractice\test_C++\helloworldcpp\cin> g++ convert.cpp -o convert
PS C:\vscodepractice\test_C++\helloworldcpp\cin> ./convert.exe
Enter Capital or Small letter:
a
input: a
output: A
PS C:\vscodepractice\test_C++\helloworldcpp\cin>
```

```
void convertAlphabet(char alphabet)
{

  if (alphabet >= 'a' && alphabet <= 'z') // Convert lowercase to uppercase
  {
    cout << "input: ";
    cout << alphabet << endl;
    cout << "output: ";
    alphabet = alphabet - 32;
    cout << alphabet << endl;
  }
  else if (alphabet >= 'A' && alphabet <= 'Z') // Convert uppercase to lowercase
  {
    cout << "input: ";
    cout << alphabet << endl;
    cout << "output: ";
    alphabet = alphabet + 32;
    cout << alphabet << endl;
  }
}
```

C++ Class

- Class의 정의
  - 변수들과 연관된 함수들을 결합시킨 새로운 형
  - 클래스를 선언함으로써 새로운 타입 생성
  - 구조체를 선언하는 것도 새로운 타입을 생성하는 것이지만 **함수**까지 결합 한다는 것이 구조체와 클래스의 차이

C++ Class

- 용어
  - Member variable:
    - 클래스 내의 변수
  - Member function or method:
    - 클래스 내의 함수, 객체가 무엇을 하는지를 결정
  - Object :
    - 클래스 타입을 갖는 변수

C++ Class

- 클래스 선언 Declaration of Class
- class 클래스 명

{

  Member variable;

  Member fuction;

};

Ex)

```
class Fishbread
{
        string content;
        void Wrapped();
};
```

클래스 선언 시 메모리 할당이 되는 것이 아니라 객체를 생성할 때 메모리 할당이 일어남.

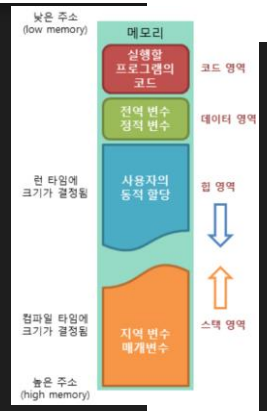
C++ 붕어빵 class (Fishbread Class)

```cpp
#include <iostream>
#include<string>
using namespace std;

class Fishbread{
    public:
    Fishbread(){
        cout<<"creat: "<<content<<endl;
    }
    Fishbread(int argCost, string argContent){
        cost = argCost;
        content = argContent;
        cout<<"creat: "<<content<<endl;
    }
    ~Fishbread(){
        cout<<"we finish eating the <"+content+"> fishbread"<<endl;
    }

    int getCost(){return cost;}
    int setCost(int c){ cost = c;}

    private:
    int cost=300;
    string content = "default";
};
```




```cpp
int main(){
    Fishbread fish1(500,"red bean");  스택에 할당
    Fishbread *fish2 = new Fishbread(600, "cream");힙에 할당
    cout<<"how much?"<<fish1.getCost()<<endl;
    fish2->setCost(400);
    cout<<"how much?"<<fish2->getCost()<<endl;

    delete(fish2);
    return 0;
}
```

이 부분이 없다면 어떻게 프린트가 되고
왜 그렇게 나올까요? (생각해보고 찾아보세요!)

Fish2는 heap에 동적할당 직접 메모리 해제를 delete를 통해 실행

main 함수의 스택을 벗어 났으므로 fishbread fish1 객체 삭제 -> fish1 객체의(인스턴스)의 소멸자

```
we finish eating the <read bean>fishbread
PS C:\vscodepractice\test_C++\helloworldcpp\cin> g++ fishbread.cpp -o fishbread
PS C:\vscodepractice\test_C++\helloworldcpp\cin> .\fishbread.exe
creat: read bean
creat: cream
how much?500
how much?400
we finish eating the <read bean>fishbread
PS C:\vscodepractice\test_C++\helloworldcpp\cin>
```

소멸자(destructor) ~Object

C++에서 생성자는 객체 멤버의 초기화뿐만 아니라, 객체를 사용하기 위한 외부 환경까지도 초기화하는 역할을 합니다. 소멸자는 객체의 수명이 끝나면 컴파일러에 의해 자동으로 호출되며, 사용이 끝난 객체를 정리해 줍니다.

C++ Class

- 접근제어 지시자 Access modifier
  - public : 어디서든 접근 허용 can be accessed everywhere
  - protected : 상속관계에 놓여 있을 때, 유도 클래스에서의 접근 허용
  - private : 클래스 내(클래스 내에 정의된 함수)에서만 접근 허용
  - 명시되지 않은 경우에 대해서는 default로 public
- 클래스에서 private은 외부로 객체의 data를 마음대로 접근할 수 없도록 하기 위해(캡슐화) 사용하며, 캡슐화된 데이터에 접근하기 위해서는 public으로 선언된 메소드를 선언

```
fish1.cost = 1000;
```

C++ - header file

- 클래스 선언과 멤버 함수 작성
  - 클래스와 사용자 간의 통신 인터페이스
  - 클래스의 자료형, 함수 종류 알림
  - *.h 파일 사용
  - ➢ 헤더 파일에는 클래스가 어떻게 생겼는지에 대한 기본 정보가 포함되어 있음
- 함수 정의
  - 함수의 구체적 동작 정의
  - *.cpp 파일 사용

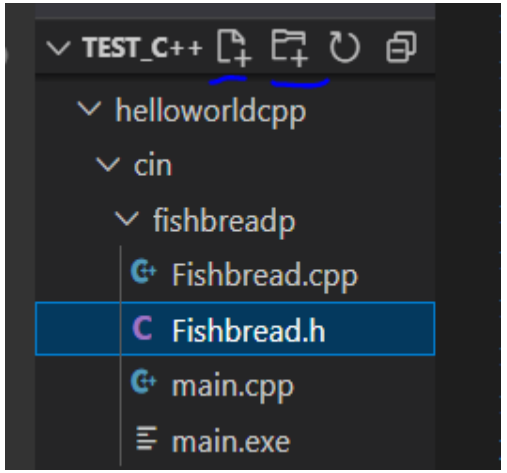
Fishbread.h

```
#include <iostream>
#include <string>
using std::string;

class Fishbread
{
public:
    Fishbread();
    Fishbread(int argCost, string argCtent);
    ~Fishbread();
    int getCost();
    void setCost(int c);

private:
    int cost;
    string content;
};
```

Exercise 3: 오버로딩 문제



- 한 붕어빵 클래스 내에 같은 이름의 생성자를 여러 개 정의(Fishbread.cpp)
  - FishBread() 생성자는 content는 unknown fishbread이며 cost가 100원
  - Fishbread (int argCost) 생성자는 content는 nobrand fishbread
  - Fishbread (string argContent) 생성자는 cost가 150원
  - Fishbread (int argCost, string argContent);
  - ~Fishbread()소멸자는 we finish eating the <content> fishbread를 출력
  - setCost 함수는 받은 매개변수값을 cost로 변경

```
PS C:\vscodepractice\test_C++\helloworldcpp\cin\fishbreadp> g++ main.cpp -o main
PS C:\vscodepractice\test_C++\helloworldcpp\cin\fishbreadp> ./main
how much and what fishbread1? 100 unknown fishbread
how much and what fishbread1? 800 unknown fishbread
how much and what fishbread2? 150 white
how much and what fishbread3? 300 nobrand fishbread
how much and what fishbread4? 500 read bean

we finish eating the <read bean> fishbread
we finish eating the <nobrand fishbread> fishbread
we finish eating the <white> fishbread
we finish eating the <unknown fishbread> fishbread
PS C:\vscodepractice\test_C++\helloworldcpp\cin\fishbreadp>
```

출력 화면

```h
#include <iostream>
#include <string>
using std::string;
class Fishbread
{
    public:
    Fishbread ();
    Fishbread (int argCost);
    Fishbread (string argContent);
    Fishbread (int argCost, string argContent);
    ~Fishbread();

    int getCost();
    void setCost(int c);
    string getContent();

    private:
        int cost;
        string content;
};
```

Fishbread.h

```cpp
#include "Fishbread.cpp"
using namespace std;

int main()
{
    Fishbread fish1;
    Fishbread fish2("white");
    Fishbread fish3(300);
    Fishbread fish4(500, "read bean");

    cout<< "how much and what fishbread1? "<< fish1.getCost()<<" "<<fish1.getContent()<<endl;
    fish1.setCost(800);
    cout << "how much and what fishbread1? "<< fish1.getCost()<<" "<<fish1.getContent()<<endl;
    cout << "how much and what fishbread2? "<< fish2.getCost()<<" "<<fish2.getContent()<<endl;
    cout << "how much and what fishbread3? "<< fish3.getCost()<<" "<<fish3.getContent()<<endl;
    cout << "how much and what fishbread4? "<< fish4.getCost()<<" "<<fish4.getContent()<<endl;
    cout<<endl;

    return 0;
}
```

main.cpp

- Fishbread.cpp를 작성하기

```cpp
#include "Fishbread.h"
using std::cout;
using std::cin;
using namespace std;


    Fishbread::Fishbread(){
            //fill in
    }
    Fishbread::Fishbread(int argCost, string argContent)
    {
       //fill in
    }

    Fishbread::Fishbread(int argCost)
    {
       //fill in
    }
    Fishbread::Fishbread(string argContent)
    {
       //fill in
    }
    Fishbread::~Fishbread()
    {
       //fill in
    }
    int Fishbread::getCost()
    {
        return cost;
    }

    void Fishbread::setCost(int c)
    {
        //fill in
    }
     string Fishbread::getContent()
     {
        return content;
     }
```

Fishbread.cpp