C++ 스터디 1주차

C++

- “절차 지향적 언어” vs. “객체 지향적 언어”

객체?

- 객체 지향 언어를 이루는 기본 요소

- 자세한 내용은 앞으로 할 것

표준 입출력(C)

stdio.h printf, scanf

- C++에서 C의 라이브러리 함수 사용 가능(stdio.h 등 이용 가능)

- 단, C++만의 라이브러리와 호환을 위해선 cstdio로 변경

(추후 다시 설명)

표준 입출력(C++)

iostream cout, cin

- input/output stream (뒤에 .h가 붙지 않음)

- cout, cin은 iostream 헤더에 정의된 표준 입출력 객체

- <<, >> 연산자와 함께 사용 (연산자 오버로딩에서 더 자세히 설명)

연속해서 사용 가능

표준 입출력(C++)

```cpp
#include<iostream>

int main(void)
{
    std::cout << "Hello, World!" << std::endl;

    return 0;
}
```

Namespace

std

- 서로 다른 라이브러리끼리 함수, 변수 등의 이름 충돌 가능성

-> namespace 이용

- 서로 다른 namespace끼리는 이름 겹치기 가능

Namespace

```cpp
#include<iostream>

namespace Woo
{
    void Hello()
    {
        std::cout << "Hi" << std::endl;
    }
}

namespace Yeol
{
    void Hello()
    {
        std::cout << "Bye" << std::endl;
    }
}
```

```cpp
int main(void)
{
    Woo::Hello();
    Yeol::Hello();

    return 0;
}
```

Namespace

using ~~~

- 1. 특정 namespace에 포함된 식별자를 global namespace에 있는 것처럼 이용

ex) using Woo::Hello;

- 2. 특정 namespace에 포함된 모든 식별자를 global namespace에 있는 것처럼 이용

ex) using namespace Woo;

Namespace

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

int main(void)
{
    char name[10];

    cout << "Enter name : ";
    cin >> name;

    cout << "Hello, " << name << endl;

    return 0;
}
```

```
[          @localhost study]$ a.out
Enter name : wooyeol
Hello, wooyeol
```

Namespace

stdio.h **cstdio**

- **stdio.h는 namespace 없음**

- **cstdio는 std namespace에 존재**

덧셈 프로그램

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

int main(void)
{
    int x, y;

    cout << "두 수 입력 : ";
    cin >> x >> y;

    cout << "합 : " << x + y << endl;

    return 0;
}
```

함수 오버로딩

- 같은 이름을 가진 여러 개의 함수 존재 가능

- 조건 : 매개 변수가 달라야 함(개수, 타입)

함수 오버로딩

```
 1 #include<iostream>
 2 using namespace std;
 3
 4 int sum(int x, int y)
 5 {
 6     cout << "int sum" << endl;
 7     return x + y;
 8 }
 9
10 double sum(double x, double y)
11 {
12     cout << "double sum" << endl;
13     return x + y;
14 }
```

```
16 int main(void)
17 {
18     cout << sum(1, 2) << endl;
19
20     cout << sum(1.2, 3.5) << endl;
21
22     return 0;
23 }
```

```
[          @localhost study]$ a.o
int sum
3
double sum
4.7
```

default parameter

- 매개변수의 기본값

- 값을 전달하지 않은 경우, 그 값을 가짐

default parameter

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

int sum(int x, int y = 3)
{
    return x + y;
}

int main(void)
{
    cout << sum(3) << endl;

    cout << sum(4, 6) << endl;

    return 0;
}
```

```
[                    )localhost study]$ a.out
6
10
```

default parameter

- 뒤에 있는 매개변수부터 줄 수 있음

- 함수 프로토타입이나 정의 둘 중 하나에만 기술(일반적으로 프로토타입에 기술)

메모리 동적 할당

new, delete

- 각각 malloc과 free에 대응

메모리 동적 할당

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

int main(void)
{
    int* ip = new int;

    *ip = 3;

    cout << *ip << endl;

    delete ip;

    return 0;
}
```

```cpp
#include<iostream>
#include<cstdlib>
#include<ctime>
using namespace std;

int main(void)
{
    srand((unsigned)time(NULL));

    int* arr = new int[10];

    for(int i = 0;i < 10;i++)
        arr[i] = rand() % 100;


    for(int i = 0;i < 10;i++)
        cout << arr[i] << ' ';
    cout << endl;

    delete[] arr;

    return 0;
}
```

bool type

- 조건 검사에 사용하는 타입
- 정수와 덧셈 가능 (true : 1, false : 0 취급)
- 1 또는 0만 저장 가능

ex) bool a = 3; //a에 1 저장됨

inline function

- 기술은 함수처럼, 작동은 매크로처럼

inline function

```
 1 #include<iostream>
 2 using namespace std;
 3 #define SUM(X, Y) (X)+(Y)
 4
 5 int main(void)
 6 {
 7     cout << SUM(1, 3) << endl;
 8
 9     return 0;
10 }
```

```
 1 #include<iostream>
 2 using namespace std;
 3
 4 inline int SUM(int x, int y)
 5 {
 6     return x + y;
 7 }
 8
 9 int main(void)
10 {
11     cout << SUM(1, 3) << endl;
12
13     return 0;
14 }
```