

# | type casting, 빠른 입출력 data의 크기 |

광운대학교  
소프트웨어학부

**김석희**

2023.09.02

# data type의 크기

정수형	signed char	8bit 이상, short 이하 (일반적으로 8bit)	-128~127	실수형	float	char 이상 (일반적으로 32bit)	3.4E+/-38(7개의 자릿수)
	unsigend char		0~255				
	signed short	16bit 이상, char 이상, int 이하 (일반적으로 16bit)	-32,767~32,767				
	unsigned short		0~65,535				
	signed int	16bit 이상, short 이상 (일반적으로 32bit)	-2,147,483,648~2,147,483,647				
	unsigned int		0~4,294,967,295				
	signed long	32bit 이상, int 이상 (일반적으로 32bit)	-2,147,483,648~2,147,483,647				
	unsigned long		0~4,294,967,295				
	signed long long	64bit 이상, int 이상 (일반적으로 64bit)	-9,223,372,036,854,775,808~9,223,372,036,854,775,807				
	unsigned long long		0~18,446,744,073,709,551,615				

실수형	float	char 이상 (일반적으로 32bit)	3.4E+/-38(7개의 자릿수)
	double	float 이상 (일반적으로 64bit)	1.7E+/-308(15개의 자릿수)
	long double	double 이상 (일반적으로 64bit)	double과 동일하나 정밀도가 더 높음

## data type의 크기

signed int	16bit 이상, short 이상 (일반적으로 32bit)	-2,147,483,648~2,147,483,647
unsigned int		0~4,294,967,295
signed long long	64bit 이상, int 이상 (일반적으로 64bit)	-9,223,372,036,854,775,808~9,223,372,036,854,775,807
unsigned long long		0~18,446,744,073,709,551,615

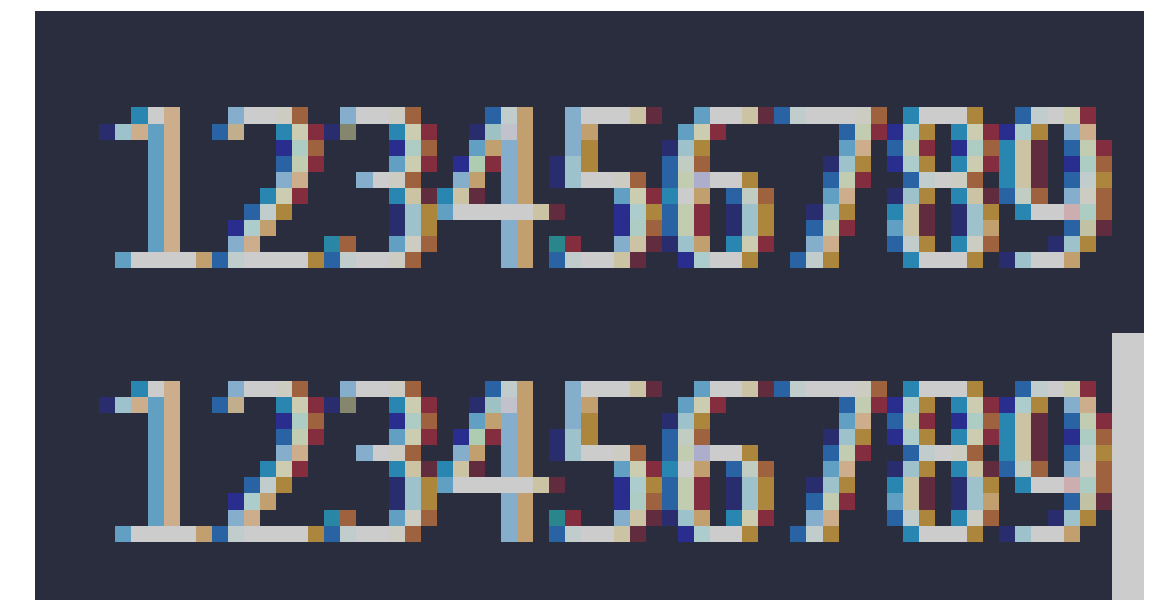
```
#include <iostream>

using namespace std;

int main() {

    int a = 123456789;
    long long b = 123456789;

    cout << a << "\n" << b;
    return 0;
}
```



```
123456789
123456789
```

## data type의 크기

signed int	16bit 이상, short 이상 (일반적으로 32bit)	-2,147,483,648~2,147,483,647
unsigned int		0~4,294,967,295
signed long long	64bit 이상, int 이상 (일반적으로 64bit)	-9,223,372,036,854,775,808~9,223,372,036,854,775,807
unsigned long long		0~18,446,744,073,709,551,615

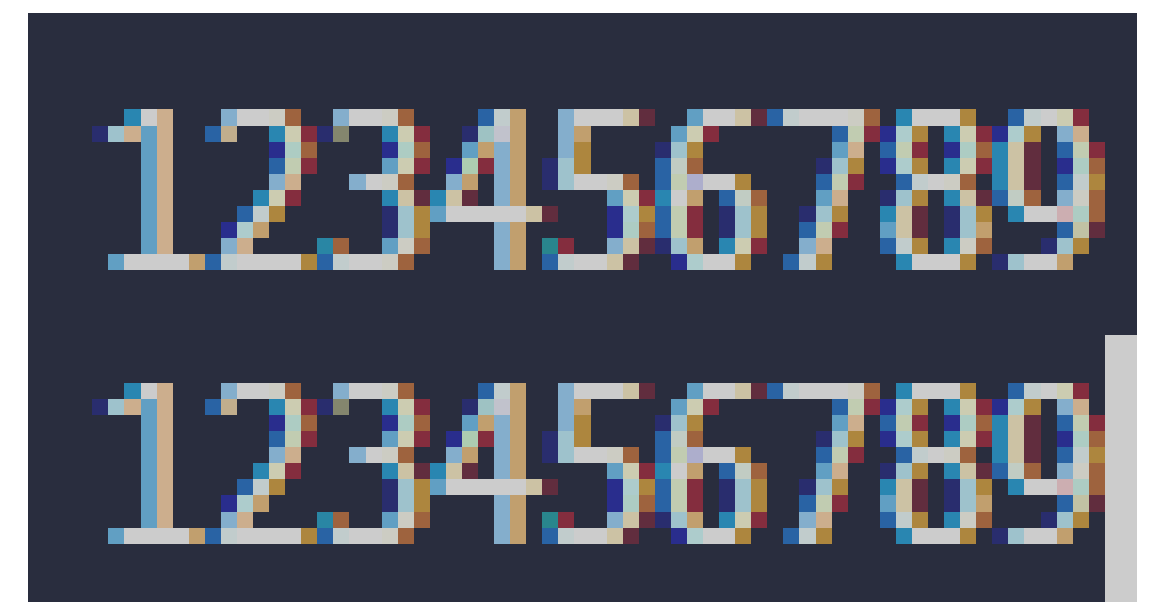
```
#include <iostream>

using namespace std;

int main() {

    int a = 123456789;
    long long b = 123456789;

    cout << a << "\n" << b;
    return 0;
}
```



```
123456789
123456789
```

# data type의 크기

signed int	16bit 이상, short 이상 (일반적으로 32bit)	-2,147,483,648~2,147,483,647
unsigned int		0~4,294,967,295
signed long long	64bit 이상, int 이상 (일반적으로 64bit)	-9,223,372,036,854,775,808~9,223,372,036,854,775,807
unsigned long long		

```
#include <iostream>

using namespace std;

int main() {

    int a = 123456789123456789;
    long long b = 123456789123456789;

    cout << a << "\n" << b;
    return 0;
}
```

```
C:\Users\JH\Documents\K_SukH\algorithm\algo_practice\tndjq.cpp:7:13: warning: overflow in implicit constant conversion [-Woverflow]
    int a = 123456789123456789;
            ^~~~~~
```

## data type의 크기

signed int	16bit 이상, short 이상 (일반적으로 32bit)	-2,147,483,648~2,147,483,647
unsigned int		0~4,294,967,295
signed long long	64bit 이상, int 이상 (일반적으로 64bit)	-9,223,372,036,854,775,808~9,223,372,036,854,775,807
unsigned long long		0~18,446,744,073,709,551,615

```
#include <iostream>

using namespace std;

int main() {

    long long a = 123456789123456789;
    long long b = 123456789123456789;

    cout << a << "\n" << b;
    return 0;
}
```

```
123456789123456789
123456789123456789
```

# type casting

## 형변환시 고려사항

큰 자료형에서 작은 자료형으로 형변환을 할 경우 일부가 잘려나갈 수 있다.

```
001 #include "stdafx.h"
002 #include <iostream>
003
004 using namespace std;
005
006 int _tmain(int argc, _TCHAR* argv[])
007 {
008     float number1 = 55.55;
009     int number2 = (int)number1;
010     bool number3 = (bool)number1;
011     cout << number2 << "\n";
012     return 0;
013 }
```



포스트 링크 <[\[C++ 정리\] 자료형의 크기 및 범위](#)>에 따라 55.55는 각각 55(int), 1(bool)이 출력된다.

# type casting

## 명시적 형변환

```
001 float number1 = 55.55;  
002 int number2 = (int)number1;  
003 bool number3 = (bool)number1;
```

?

변환할 자료형을 명시((int))해준다.

## 묵시적 형변환

```
001 float number1 = 55.55;  
002 int number2 = number1;  
003 bool number3 = number1;
```

?

그냥 대입한다.

## 명시적 형변환과 묵시적 형변환의 차이점

- \* 결과적인 차이는 없다
- \* 명시적 형변환의 경우 내부적으로 임시변수를 생성에 대입하는 방식으로 성능 저하를 일으킬 수 있다.
- \* 묵시적 형변환의 경우 데이터 손실에 대한 경고가 발생한다.



string 에서 int로, int에서 string으로

```
#include <string>
```

```
int stoi(const string& str, size_t* idx = 0, int base = 10);
```

# string 에서 int로, int에서 string으로

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

int main() {
    string a = "1234", b = "1a2b3c";
    int num = 4321;

    // 1. string -> int
    cout << stoi(a) << "\n";

    // 2. string 문자열에서 숫자만 구분하여 출력
    for (int i = 0; i < b.size(); i++) {
        if (b[i] >= '0' && b[i] <= '9')
            cout << b[i] - '0';
    }
    cout << "\n";

    // 3. int -> string
    string s = to_string(num);
    string r = "result = ";

    r += num;
    cout << r << "\n";

    r = "result = ";
    r += s;
    cout << r << "\n";

    return 0;
}
```

```
1234
123
result =
result = 4321
```

string 에서 int로, int에서 string으로

```
// 1. string -> int
```

```
cout << stoi(a) << "\n";
```

## string 에서 int로, int에서 string으로

```
int num = 4321;
```

```
// 3. int -> string
```

```
string s = to_string(num);
```

```
string r = "result = ";
```

```
r += num;
```

```
cout << r << "\n";
```

```
r = "result = ";
```

```
r += s;
```

```
cout << r << "\n";
```

```
return 0;
```

**빠른 입출력**

**boj.kr/15552**

## 빠른 입출력

```
ios_base::sync_with_stdio(false);  
cin.tie(nullptr);
```

## 빠른 입출력

`ios_base::sync_with_stdio(false);`

```
#include <iostream>

using namespace std;

int main() {
    cout << "20";
    printf("1");
    return 0;
}
```



## 빠른 입출력

`ios_base::sync_with_stdio(false);`

```
#include <iostream>

using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cout << "20";
    printf("1");
    return 0;
}
```

120



## 빠른 입출력

```
cin.tie(NULL);  
cout.tie(NULL);
```

`cin.tie(nullptr), cout.tie(nullptr)`

**원래 묶여있음. 이유는 순서를 지키기 위해서  
당연한 것은 없음.**

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

버퍼가 시간을 잡아먹기 때문에,  
입출력의 변환이 빈번하게 이루어지는 경우  
`untie`를 하게 되면 더욱 입출력이 빨라지게 됨

## 빠른 입출력

15552 - `sync_with_stdio(false)`, `cin.tie(nullptr)`

11021, 11022 - 문제의 요구대로 출력하는 법

10953, 2577 - 적절한 데이터 타입을 고르는 법

2442, 2523, 2522 - 별찍기 시리즈(5,13,12)

1110, 2869, 1475, 1193 - 구현 (생각해야 하는 문제들)

2학년 학생들은 구현 두문제 이상 풀고 코드 올려볼 것

<https://padlet.com/s2k616/2023-fqhzayk4f2panibl>

## 과제 제출 방법

<https://padlet.com/s2k616/2023-fqhzayk4f2panibl>

