# 창의적 소프트웨어 설계

2주차 실습 – 메모리 기초

노인우, inwoo13@hanyang.ac.kr
한중수, wndtnsla@hanyang.ac.kr

# Overview

## 목표

- C/C++기본 자료형, 연산자, 포인터, 동적 메모리 할당 실습
- 기본 입출력 실습
- 메모리 관점에서 이해

# 기본 자료형

- Integer
  - int
  - char (문자형은 정수형의 한 가지)
  - short
  - long
  - long long
  - unsigned
- Boolean
  - bool
- Floating Point(부동 소수점)
  - float
  - double

# 기본 자료형 – 크기

## ◆ 자료형 크기 확인 방법

- sizeof() 활용 예제

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

int main(){
	int iVar = 10;
	float fVar = 10;
	double dVar = 10;

	// 변수 크기 확인
	cout << "Size of int: " << sizeof(iVar) << endl;
	cout << "Size of float: " << sizeof(fVar) << endl;
	cout << "Size of double: " << sizeof(dVar) << endl;

	// 자료형 크기 확인
	cout << "Size of long: " << sizeof(long) << endl;
}
```

# 기본 자료형 – 크기

## ◆ 자료형 크기 확인 방법

- sizeof() 활용 예제 결과

```
imtutor@imtutor-desktop:~/class_materials/0913$ g++ -o week2 example.cpp
imtutor@imtutor-desktop:~/class_materials/0913$ ./week2
Size of int: 4
Size of float: 4
Size of double: 8
Size of long: 8
imtutor@imtutor-desktop:~/class_materials/0913$
```

# 기본 자료형 – C++ 상수

◆ const int 변수 값 변경 시도

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

int main(){
	const int TEST_NUM = 10;
	TEST_NUM = 3;

	cout << "Result: " << TEST_NUM << endl;
}
```

# 기본 자료형 – C++ 상수

- const int 변수 값 변경 시도 결과

```
imtutor@imtutor-desktop:~/class_materials/0913$ g++ -o week2 example.cpp
example.cpp: In function ‘int main()’:
example.cpp:6:11: error: assignment of read-only variable ‘TEST_NUM’
  TEST_NUM = 3;
           ^
imtutor@imtutor-desktop:~/class_materials/0913$
```

# 연산자 (Operator)

- Increment / Decrement:
  - a++, ++a
  - a--, --a
- Relational:
  - a == b, a != b
  - a < b, a <= b, a > b, a >= b
- Bitwise:
  - a & b, a | b, a ^ b, ~a, a >> b, a << b
- Logical :
  - a && b, a || b, !a
- 우선순위에 확신이 없을 시 () 괄호 활용!

# 연산자 (Operator) – 우선순위

◆ 예시

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

int main(){
	int a = 10;

	cout << "Result: " << ++a << endl;
	cout << "Result: " << a++ << endl;
}
```

# 연산자 (Operator) – 우선순위

## ◆ 예시 결과

```
imtutor@imtutor-desktop:~/class_materials/0913$ make
g++ -c -o week2.o example.cpp
g++ -o week2 week2.o
./week2
Result: 11
Result: 11
```

# C++ 문자열 입출력

- C++ strings
  - char 배열에 비해서 다양한 메소드 지원
- C++ iostream
  - std::cin
  - std::cout

# C++ 문자열 입출력

## ◆ 예시

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

int main(){
	string str_empty;
	string str_init1("hey!! ");
	string str_init2 = "another way to init!";

	cout << "Ok! " << str_init1 << str_init2 << endl;
}
```

# C++ 문자열 입출력

## ◆ 예시 결과

```
g++ -c -o week2.o example.cpp
g++ -o week2 week2.o
./week2
Ok! hey!! another way to init!
```

# 포인터 (Pointer)

## ◆ 메모리 주소를 저장하는 변수

- 다른 변수, 함수의 시작 주소를 저장

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

int main(){
    int a = 10;
    int* pa = &a;

    cout << "What is in a: " << a << endl;
    cout << "What is a's address: " << &a << endl;
    cout << "What is in pa: " << pa << endl;
    cout << "What is in pa's target: " << *pa << endl;
}
```

```
What is in a: 10
What is a's address: 0x7ffd5e29a83c
What is in pa: 0x7ffd5e29a83c
What is in pa's target: 10
```

# 포인터 (Pointer)

- Call-by-Value
- Call-by-Reference

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

int func_cbv(int a_in){
   a_in++;
}

int func_cbr(int* pa_in){
   (*pa_in)++;
}
```

```
int main(){
      int a = 10;
      int *pa = &a;

      func_cbv(a);
      cout << "Result: " << a << endl;

      func_cbr(pa);
      cout << "Result: " << a << endl;
}
```

# 포인터 (Pointer) - 예제

## ◆ 포인터 크기 확인

- 64bit 환경에서는 Size of pointer: 8
- 32bit 환경에서는 Size of pointer: 4

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

int chkPointerSize(){
   cout << "Size of pointer: " << sizeof(void*) << endl;
}

int main(){
      chkPointerSize();
}
```

# 포인터 (Pointer) - 예제

## ◆ 포인터 크기 확인

- 64bit 환경에서는 Size of pointer: 8
- 32bit 환경에서는 Size of pointer: 4

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

int chkPointerSize(){
   cout << "Size of pointer: " << sizeof(void*) << endl;
}

int main(){
      chkPointerSize();
}
```

# C 동적 메모리 할당 (malloc / free)

- malloc
  - 동적 메모리 할당
- free
  - 동적으로 할당된 메모리를 반환
- 메모리 누수
  - 동적으로 메모리를 할당하고, 반환하지 않는 로직
  - Heap 메모리 영역이 팽창

# C++ 동적 메모리 할당 (new / delete)

- new / delete는 C++에서 동적 메모리 할당 키워드

- malloc/free와의 차이점:

| | malloc / free | new / delete |
|---|---|---|
| 종류 | 함수 | 키워드 |
| 할당 타입 | 기본적으로 void* | 지정 타입 |
| 초기값 | 지정 불가 | 지정 가능 |
| 클래스 생성자 | - | 자동 호출 |

# C++ 동적 메모리 할당 (new / delete)

## ◆ 예제

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

int main(){
    int array_size = 10;
        int* a = new int[array_size](); // init to zero
        int* b = new int[array_size]{1, 2, 3, 1, 2, 3, 1, 2, 3, 0}; // init in manual

        for(int n = 0 ; n < array_size ; n++){
            cout << "Index[" << n << "] Data: " << a[n] << endl;
            cout << "Index[" << n << "] Data: " << b[n] << endl;
            cout << "-----------------------------------" << endl;
        }
}
```

# C++ 동적 메모리 할당 (new / delete)

## ◆ 예제 결과

```
Index[0] Data: 0
Index[0] Data: 1
-------------------------------------
Index[1] Data: 0
Index[1] Data: 2
-------------------------------------
Index[2] Data: 0
Index[2] Data: 3
-------------------------------------
Index[3] Data: 0
Index[3] Data: 1
-------------------------------------
Index[4] Data: 0
Index[4] Data: 2
-------------------------------------
Index[5] Data: 0
Index[5] Data: 3
```

# 참고자료

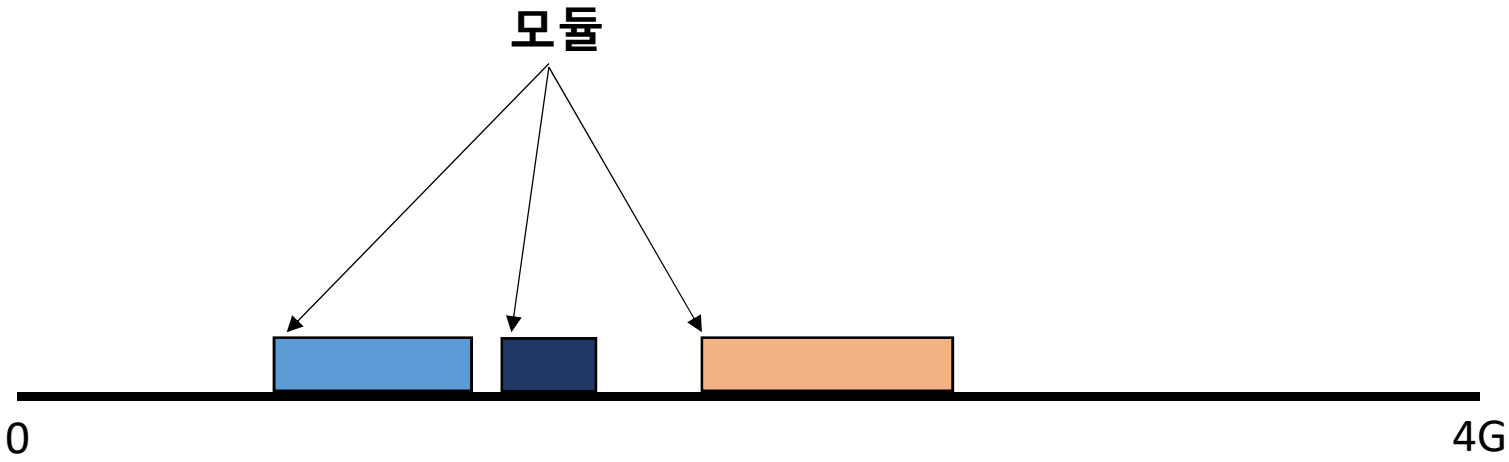

1. Windows 버전 별 한계 메모리, https://msdn.microsoft.com/ko-kr/library/windows/desktop/aa366778(v=vs.85).aspx
2. Ubuntu 64-bit 한계 메모리, https://help.ubuntu.com/community/32bit_and_64bit
3. int / long 차이, http://smallpants.tistory.com/10
4. 네임스페이스, http://thinkpro.tistory.com/22


Q&A

# Appendix #1. Memory

## ◆ 1차원 주소 공간

- 함수, 변수(전역, 지역, 동적) 모두 메모리 위에 존재



# Appendix #2. Heap vs Stack

- Heap
  - 동적 메모리 할당
- Stack
  - 정적 변수 메모리
  - 정확히는 함수
- 메모리 구조가 다른 것
- 메모리 공간 자체가 다른 것은 아니다

# Appendix #3. 부동소수점 변수?

## ◆ 부동소수점 변수의 유효 자리수

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

int main(){
        int a = 10;
        float fval = 1234.123456789;
        double dval1 = 1234.123456789;

    cout.precision(10);
        cout << "float: " << fval << endl;
        cout << "double: " << dval1 << endl;
}
```

```
float: 1234.123413
double: 1234.123457
```

# Appendix #4. Namespace

- 함수, 변수 등의 이름이 어디에 소속되어 있는지!

- 다음은 모두 다른 내용
  - std::test_value
  - tutor::test_value
  - student::test_value

# Appendix #5. Make - Build & Run

## make all

```
week2: week2.o
	g++ -o week2 week2.o
	./week2

week2.o: example.cpp
	g++ -c -o week2.o example.cpp

clean:
	rm *.o week2

all:
	make clean
	make
```

# Appendix #6. Pointer – Buffer Overflow

## Buffer Overflow

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

int main(){
    char buffer[10];

    cin >> buffer;
    cout << "What's in buffer: " << buffer << endl;
}
```

```
imtutor@imtutor-desktop:~/class_materials/0913$ ./week2
11111111111111111111111111111111111111111111111111
What's in buffer:
11111111111111111111111111111111111111111111111111
*** stack smashing detected ***: ./week2 terminated
Aborted (core dumped)
```