

자료구조 & 알고리즘

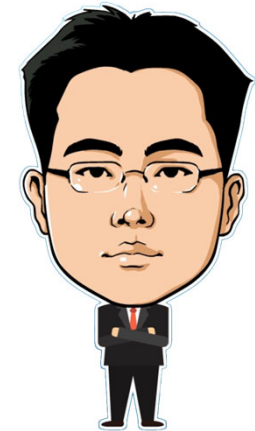
for(A;B;C)
D;



C++ 프로그래밍
(C++ Programming)

Seo, Doo-Ok

Clickseo.com
clickseo@gmail.com



목 차



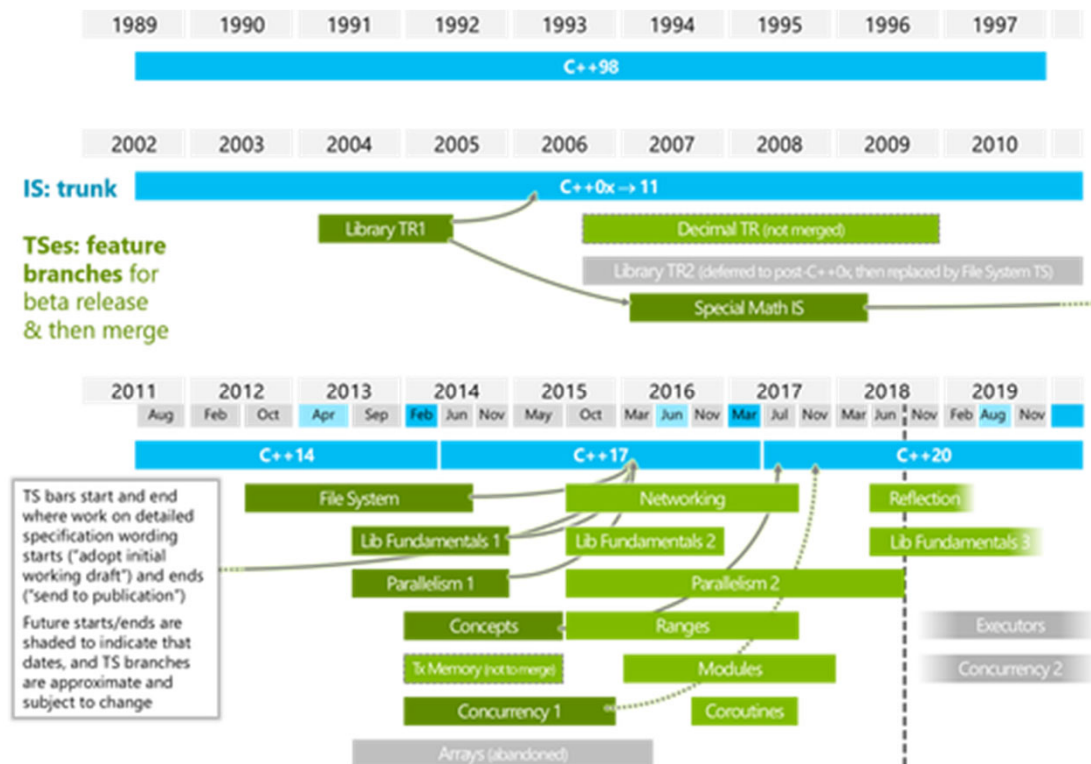
- C++ 프로그래밍 기초
- 객체지향 프로그래밍



C++ 언어 개요

- C++ 언어 표준화

- 2018년 06월, ISO C++ standards meeting, "WG21 timeline"



[출처 : Herb Sutter, "Trip report : Summer ISO C++ standards meeting", 2018.]

C++ 프로그래밍 기초



- C++ 프로그래밍 기초

- C++ 프로그램 구조

- 배열, 문자열, 구조체

- 함수, 네임스페이스, 참조

- 동적 메모리 할당 ex) *list* ...

- 객체지향 프로그래밍



C++ 프로그래밍 기초

(정정정 정정정 정정정)

C++ 프로그램 구조



C++ 프로그램 구조 (1/4)

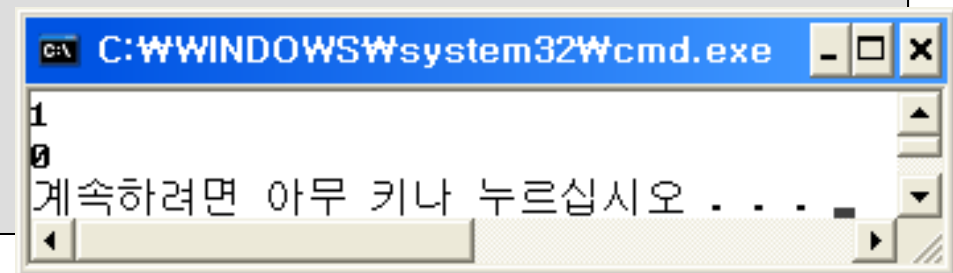
- 새로운 형태의 자료형: **bool** (C → #include <stdbool.h>)

○ bool형 변수의 상태는 **true**와 **false** 둘 중 하나가 될 수 있다.

```
#include <iostream>

int main(void)
{
    std::cout << true << std::endl;
    std::cout << false << std::endl;

    return 0;
}
```



C++ 프로그램 구조 (2/4)

● 변수(variable)

○ 변수 이름의 길이에 제한이 없다.

- C 에서 변수 이름의 길이 제한: 63번째 문자까지만 인식

○ “지역 변수 선언의 위치 제한이 없다.” → C도 켜.

```
#include <iostream>

int main(void)
{
    int    i = 100;

    std::cout << i << std::endl;

    int    j = 200;

    std::cout << j << std::endl;

    return 0;
}
```

in pg ⇒

$\begin{pmatrix} a=10 & 1000 & | & 101 \\ a \rightarrow & & & \\ a=20 & 1000 & | & 201 \end{pmatrix}$

메모리 주소 X
주소 0

C++ ⇒

$\begin{pmatrix} \text{int } a=10 & 1000 & | & 101 \\ a=20 & 1000 & | & 201 \end{pmatrix}$

C++ 프로그램 구조 (3/4)

프로그램 예제 : 데이터 입력 및 출력

```
#include <iostream>

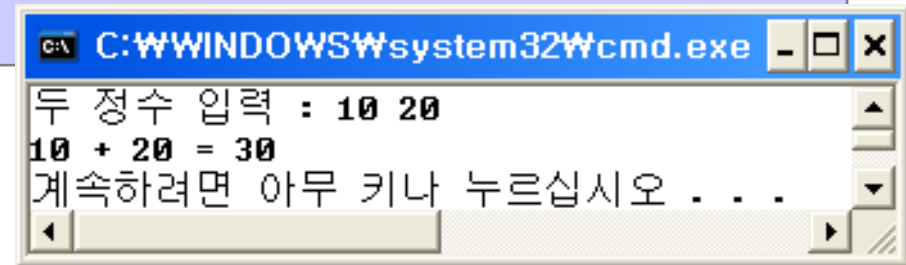
int main(void)
{
    int    a, b, res;

    std::cout << "두 정수 입력 : " ;
    std::cin >> a >> b;

    res = a + b;

    std::cout << a << " + " << b << " = " << res << std::endl;

    return 0;
}
```



C++ 프로그램 구조 (4/4)

- 명시적 형 변환(explicit type conversion)

- cast 수식 연산자(cast expression operator)

- 임의로 어떤 형식에서 다른 형식으로 데이터를 변환시킨다.

```
#include <iostream>

int main(void)
{
    int    i;
    double d = 3.14159;

    // i = (int)d;
    i = int(d);

    std::cout << "i  : " << i << std::endl;
    std::cout << "d : " << d << std::endl;

    return 0;
}
```

// C++ 에서만 가능

C++ 프로그래밍 기초

배열, 문자열, 구조체



배열, 문자열, 구조체 (1/3)

- C 언어 스타일의 문자열(string) *#include <string>*.

- C++ 에서 제공하는 문자열 조작 함수를 사용 : <cstring>

```
#include <iostream>
#include <cstring>

int main(void)
{
    char    str[] = "Hi~ Clickseo!!!";
    char    copy[1024];
    int     len = strlen(str);

    strcpy(copy, str);

    std::cout << "길이   : " << len << std::endl;
    std::cout << "str    : " << str << std::endl;
    std::cout << "copy   : " << copy << std::endl;

    return 0;
}
```

배열, 문자열, 구조체 (2/3)

- 구조체(structure)

- C++에서는 구조체의 태그(tag)가 곧 자료형이다.

```
#include <iostream>

struct _score {
    char    name[12];
    int     kor, eng, math, tot;
    float   ave;
};

int main(void)
{
    _score temp;

    return 0;
}
```

배열, 문자열, 구조체 (3/3)

프로그램 예제 : 구조체의 태그를 이용한 자료형

```
#include <iostream>

struct _score {
    char    name[12];
    int     kor, eng, math, tot;
    float   ave;
};

int main(void)
{
    _score temp;

    std::cout << "이름 : ";      std::cin >> temp.name;
    std::cout << "국어 : ";      std::cin >> temp.kor;
    std::cout << "영어 : ";      std::cin >> temp.eng;
    std::cout << "수학 : ";      std::cin >> temp.math;

    temp.tot = temp.kor + temp.eng + temp.math;
    temp.ave = float(temp.tot)/3;

    std::cout << "\n ### 출력 결과 ###" << std::endl;
    std::cout << temp.name << " " << temp.kor << " " << temp.eng << " "
        << temp.math << " " << temp.tot << " " << temp.ave << std::endl;

    return 0;
}
```

C++ 프로그래밍 기초

함수



함수 (1/5)

● 함수 다중 정의(Function Overloading)

- C++ 에서 함수들이 동일한 이름을 사용할 수 있는 기능
 - 단, 인자의 종류(개개 변수의 자료형이나 개수)는 달라야 한다.

```
int    ADD(int, int);  
double ADD(double, double);  
  
int main(void)  
{  
    ADD(10, 20);  
    ADD(10.5, 20.5);  
  
    return 0;  
}  
  
int    ADD(int a, int b) {  
    return a + b;  
}  
  
double ADD(double a, double b) {  
    return a + b;  
}
```

함수 (2/5)

● 디폴트 인자(Default Arguments)

○ 따로 값을 지정해주지 않은 경우에 선택하는 인자의 값

- 함수 호출 시 적당한 값을 모르는 경우에 사용
- 매번 함수를 호출할 때마다 똑같은 인자의 값을 적어주는 것을 피하는 용도로 사용
- 디폴트 인자의 제한: 디폴트 인자는 오른쪽 끝에 모여 있어야 한다.

```
#include <iostream>

int    ADD(int, int = 0);

int main(void)
{
    std::cout << ADD(10) << std::endl;
    std::cout << ADD(10, 20) << std::endl;

    return 0;
}

int    ADD(int a, int b) {
    return a + b;
}
```


함수 (3/5)

- 함수 오버로딩 vs. 디폴트 인자

```
#include <iostream>

int    ADD(int a);
int    ADD(int a, int b = 0);

int main(void)
{
    std::cout << ADD(10) << std::endl;    // error

    return 0;
}

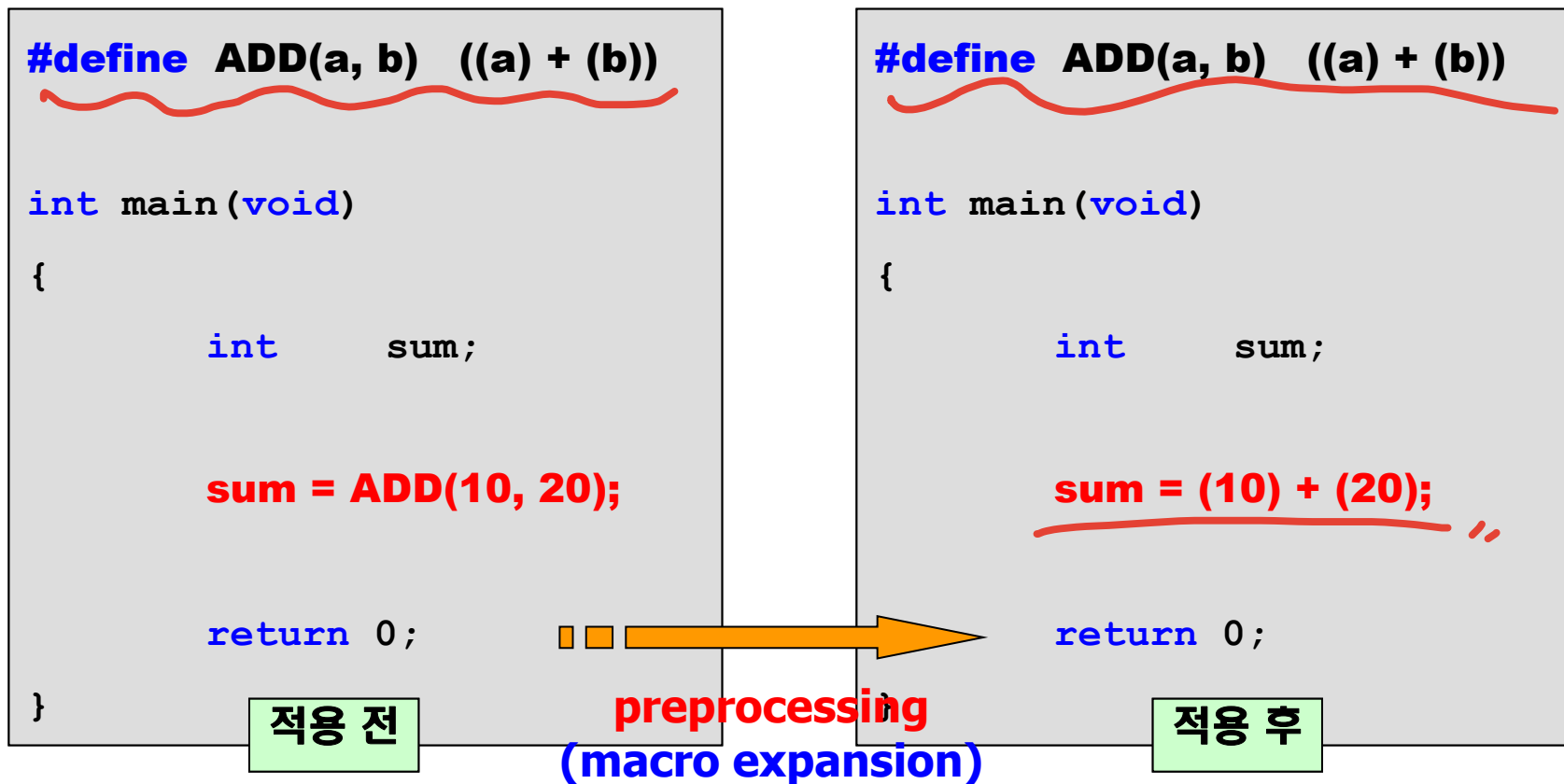
int    ADD(int a) {
    return a;
}

int    ADD(int a, int b) {
    return a + b;
}
```

함수 (4/5)

- C 언어 스타일의 in-line 화: 매크로 함수

- 프로그램을 컴파일 하기 전에 전처리에 의해 정의된 코드로 치환



함수 (5/5)

- C++ 스타일의 in-line화

- 인-라인 함수(in-line Function)
- 키워드 inline을 이용한 함수의 in-line화는 컴파일러에 의해 처리.

```
#include <iostream>
```

```
inline int ADD(int, int);
```

컴파일러에 의해 처리

```
int main(void)
```

```
{
```

```
    int    sum;
```

```
    sum = ADD(10, 20);
```

```
    std::cout << "합계 : " << sum << std::endl;
```

```
    return 0;
```

```
}
```

```
inline int ADD(int a, int b) {  
    return a + b;  
}
```

⇒ ^{같은} namespace 일기 중요.

C++ 프로그래밍 기초

네임스페이스



네임스페이스 (1/9)

- 네임스페이스의 등장 배경

- 같은 이름의 함수를 포함하면 컴파일 시 문제 발생.

```
#include <iostream>

void OUTPUT(void) {
    std::cout << "Hello World!!!" << std::endl;

    return;
}

// error C2084: function 'void __cdecl OUTPUT(void)' already has a
// body
void OUTPUT(void) {
    std::cout << "Hi~ Clickseo" << std::endl;

    return;
}

int main(void)
{
    OUTPUT();

    return 0;
}
```

⇒ 2번 2등 X → Error..

네임스페이스 (2/9)

- 네임스페이스(namespace)

- 특정 영역(공간)의 범위를 지정하고 이름을 붙여준 것

```
#include <iostream>

namespace A {
    void OUTPUT(void) {
        std::cout << "Hello World!!!" << std::endl;
        return;
    }
}

namespace B {
    void OUTPUT(void) {
        std::cout << "Hi~ Clickseo" << std::endl;
        return;
    }
}

int main(void)
{
    A::OUTPUT();
    B::OUTPUT();

    return 0;
}
```

“이름 공간이 다르면
같은 이름의 변수나 함수의 선언이 허용된다.”

범위 지정 연산자
(scope resolution operator)

네임스페이스 (3/9)

- 네임스페이스에 **별칭** 부여

- 네임스페이스의 이름이 너무 긴 경우에는 간단한 별명을 붙여준 후에, 그 별명을 대신 사용할 수 있다.

```
#include <iostream>

using std::cout;
using std::endl;

namespace Clickseo_namespace_data_temp {
    int temp;
}

namespace Click = Clickseo_namespace_data_temp;

int main(void)
{
    cout << "temp : " << Click::temp << endl;

    return 0;
}
```

네임스페이스 (4/9)

프로그램 예제 : 중첩된 네임스페이스

```
#include <iostream>

using std::cout;
using std::endl;

namespace Clickseo {
    namespace TEMP1 {
        int a = 10;
    }
    namespace TEMP2 {
        int a = 20;
    }
}

int main(void)
{
    cout << "Clickseo::TEMP1::a : " << Clickseo::TEMP1::a << endl;
    cout << "Clickseo::TEMP2::a : " << Clickseo::TEMP2::a << endl;

    return 0;
}
```


네임스페이스 (5/9)

- 이름 없는 네임스페이스

- 다른 파일에서 접근 제한

정적 (data)

dynamic (stack)
Heap
Stack

static 키워드를 사용한 전역 변수나 함수를 정의한 경우와 동일한 효과

```
#include <iostream>

using std::cout;
using std::endl;

namespace {
    int temp = 10;
}

void OUTPUT(void)
{
    cout << "temp : " << temp << endl;

    return;
}
```

static int a;
(전역)

다른 함수
전역 변수
접근

1.cpp

```
#include <iostream>

using std::cout;
using std::endl;

extern void OUTPUT(void);

extern int temp;

int main(void)
{
    cout << "temp : " << temp << endl;
    OUTPUT();

    return 0;
}
```

extern int a;
=> Error.

extern void OUTPUT(void);

extern int temp;

namespace로
붙여 주어야 함

temp
error

네임스페이스 (6/9)

- 분할 컴파일

```
namespace A { void OUTPUT(void); }
namespace B { void OUTPUT(void); }

int main(void)
{
    A::OUTPUT();
    B::OUTPUT();

    return 0;
}
```

main.cpp

```
#include <iostream>

namespace A {
    void OUTPUT(void) {
        std::cout << "Hello World!!!" << std::endl;

        return;
    }
}
```

1.cpp

```
#include <iostream>

namespace B {
    void OUTPUT(void) {
        std::cout << "Hi~ Clickseo" << std::endl;

        return;
    }
}
```

2.cpp

네임스페이스 (7/9)

프로그램 예제 : 데이터 입출력

```
#include <iostream>
```

```
int main(void)
```

```
{
```

```
    int    temp;
```

```
    std::cout << "정수 입력 : ";
```

```
    std::cin >> temp;
```

```
    std::cout << "temp : " << temp << std::endl;
```

```
    return 0;
```

```
}
```

```
namespace std
```

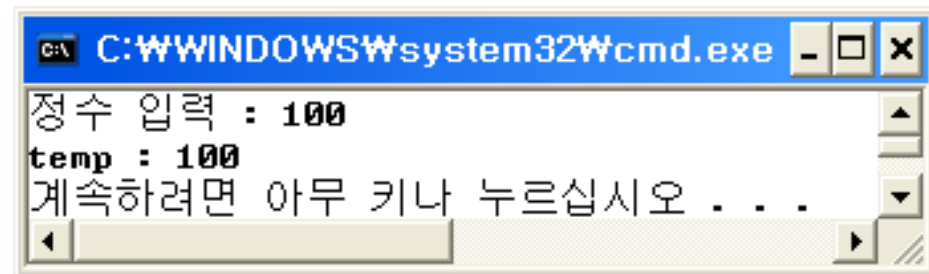
```
{
```

```
    cout    ???
```

```
    cin     ???
```

```
    endl    ???
```

```
}
```



네임스페이스 (8/9)

프로그램 예제 : 데이터 입출력 -- using

```
#include <iostream>
```

```
using std::cout;  
using std::cin;  
using std::endl;
```

```
using namespace std;
```

```
int main(void)  
{  
    int    temp;  
  
    cout << "정수 입력 : ";  
    cin >> temp;  
  
    cout << "temp : " << temp << endl;  
  
    return 0;  
}
```

네임스페이스 (9/9)

프로그램 예제 : 지역변수와 전역변수

```
#include <iostream>
```

```
using std::cout;
```

```
using std::endl;
```

```
int temp;
```

```
int main(void)
```

```
{
```

```
    int temp = 10;
```

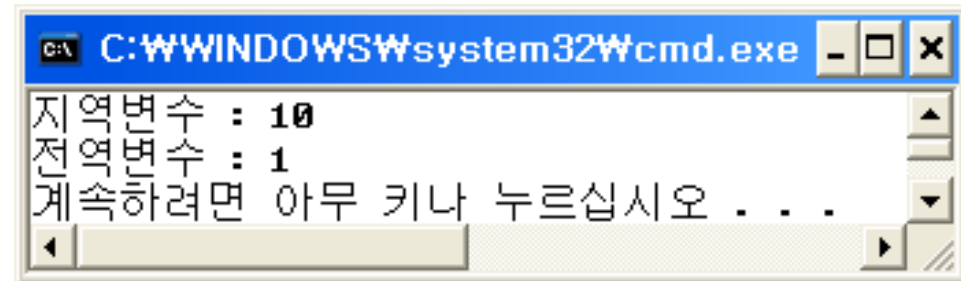
```
    ::temp++;
```

```
    cout << "지역변수 : " << temp << endl;
```

```
    cout << "전역변수 : " << ::temp << endl;
```

```
    return 0;
```

```
}
```



지역변수가 전역변수의 우선

C++ 프로그래밍 기초

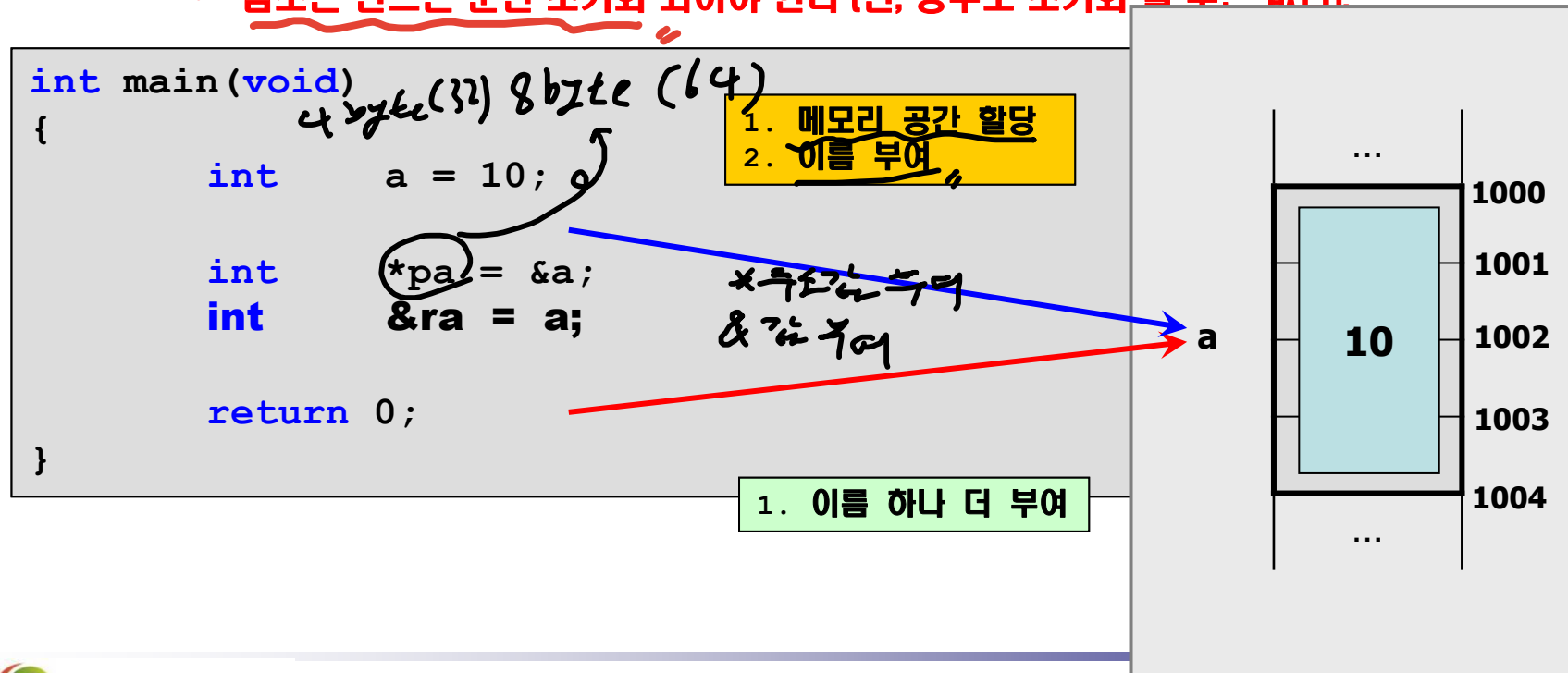
참조



참조 (1/5)

● 참조(Reference)

- 이름이 존재하는 메모리 공간에 하나의 이름을 더 부여하는 행위
 - 참조는 일종의 별칭(alias)
 - 만드는 방법 : 대상 개체의 형을 적고 참조 연산자(&)와 참조의 이름을 적으면 된다.
 - 참조는 만드는 순간 초기화 되어야 한다 (단, 상수로 초기화 할 수는 없다)**



참조 (2/5)

프로그램 예제 : 참조(reference)의 이해

```
#include <iostream>

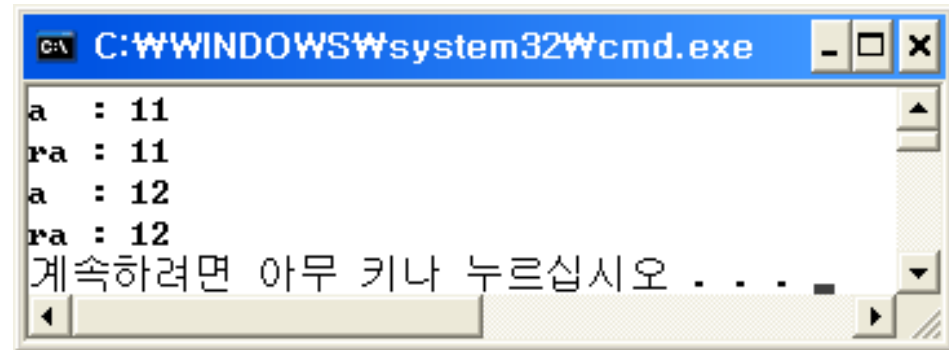
using std::cout;
using std::endl;

int main(void)
{
    int a = 10;
    int &ra = a;

    a++;
    cout << "a : " << a << endl;
    cout << "ra : " << ra << endl;

    ra++;
    cout << "a : " << a << endl;
    cout << "ra : " << ra << endl;

    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
a : 11
ra : 11
a : 12
ra : 12
계속하려면 아무 키나 누르십시오 . . .
```


참조 (3/5)

- 참조에 의한 호출(Call by reference)

```
#include <iostream>
```

```
using std::cout;  
using std::endl;
```

```
void SWAP(int &, int &);
```

```
int main(void)
```

```
{  
    int a = 10, b = 20;
```

```
    cout << "a : " << a << " , b : " << b << endl;
```

```
    SWAP(a, b);
```

```
    cout << "a : " << a << " , b : " << b << endl;
```

```
    return 0;
```

```
}
```

```
void SWAP(int &ra, int &rb) {
```

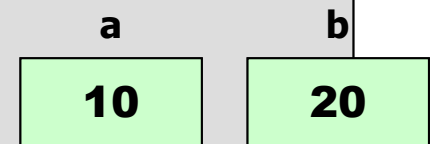
```
    int temp;
```

```
    temp = ra;
```

```
    ra = rb;
```

```
    rb = temp;
```

```
}
```



참조 (4/5)

프로그램 예제 : 부담스러운 Call-by-value

```
#include <iostream>
```

```
using std::cout;  
using std::endl;
```

```
struct _score {  
    char    name[12];  
    int     kor, eng, math, tot;  
    float   ave;  
};
```

```
void OUTPUT(_score);
```

```
int main(void)  
{
```

```
    _score data = {"서두옥", 70, 80, 90, 240, 80.0};
```

```
    OUTPUT(data);
```

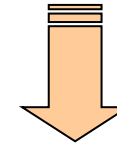
```
    return 0;
```

```
void OUTPUT(_score temp){
```

```
    cout << "이름 : " << temp.name << endl;  
    cout << "국어 : " << temp.kor << endl;  
    cout << "영어 : " << temp.eng << endl;  
    cout << "수학 : " << temp.math << endl;  
    cout << "총점 : " << temp.tot << endl;  
    cout << "평균 : " << temp.ave << endl;  
}
```

data

"서두옥"	70	80	90	240	70.0
-------	----	----	----	-----	------



"서두옥"	70	80	90	240	70.0
-------	----	----	----	-----	------

temp

byte copy

참조 (5/5)

프로그램 예제 : Call-by-reference

```
#include <iostream>

using std::cout;
using std::endl;

struct _score {
    char    name[12];
    int     kor, eng, math, tot;
    float   ave;
};

void OUTPUT(const _score &);

int main(void)
{
    _score data = {"서두옥", 70, 80, 90, 240, 80.0};
    OUTPUT(data);

    return 0;
}

void OUTPUT(const _score &temp) {
    cout << "이름 : " << temp.name << endl;
    cout << "국어 : " << temp.kor << endl;
    cout << "영어 : " << temp.eng << endl;
    cout << "수학 : " << temp.math << endl;
    cout << "총점 : " << temp.tot << endl;
    cout << "평균 : " << temp.ave << endl;
}
```

"서두옥"	70	80	90	240	70.0
-------	----	----	----	-----	------

data

temp

C++ 프로그래밍 기초

동적 메모리 할당



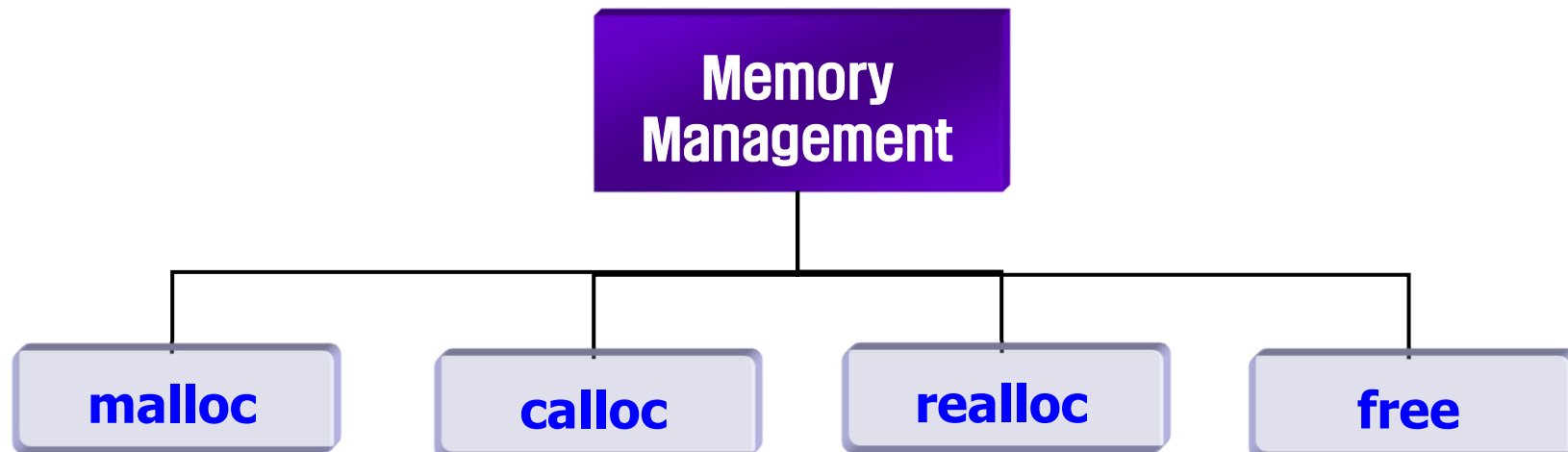
동적 메모리 할당 (1/3)

- C 언어 스타일의 동적 메모리 할당

- 동적 메모리 관리에 쓰이는 4가지 함수

- 표준 라이브러리 헤더 파일 <stdlib.h> 에서 찾을 수 있다.

- 메모리 할당 : malloc, calloc, realloc
 - 메모리 해제 : free



동적 메모리 할당 (2/3)

- C++ 스타일의 동적 메모리 할당 : new, delete 연산자

- new 연산자 : 동적 메모리 할당

- 메모리 할당 실패 시 NULL 포인터 반환

* 메모리 누수
방지

```
int *p = new int;           // 정수 하나를 저장할 메모리 할당
int *arr = new int[size];   // 정수를 size 개수만큼 저장할 메모리 할당
```

- delete 연산자 : 동적 메모리 해제

```
delete p;                   // 할당된 메모리 공간 해제
delete []arr;               // 할당된 메모리 공간이 배열일 경우
```

동적 메모리 할당 (3/3)

프로그램 예제 : 동적 메모리 할당

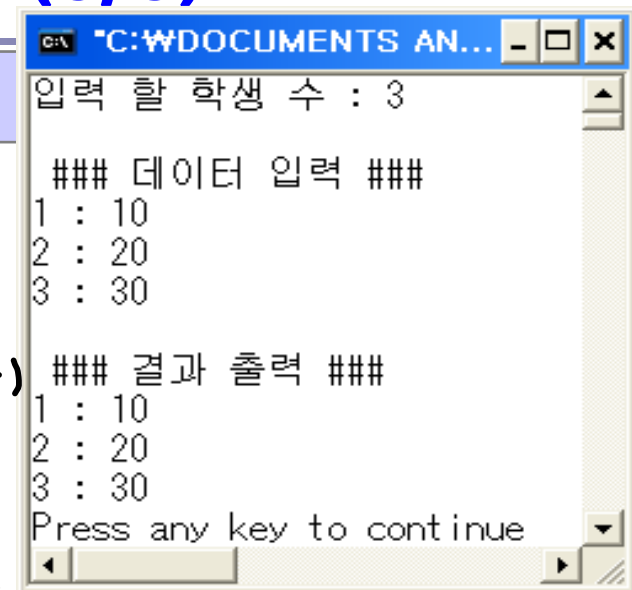
```
#include <iostream>
using namespace std;
int main(void)
{
    int i, size;
    cout << "입력 할 학생 수 : ";
    cin >> size;

    int *arr = new int[size];
    if(arr == NULL) {
        cout << "메모리 할당 실패!!!" << endl;
        return -1;
    }

    cout << "\n ### 데이터 입력 ###" << endl;
    for(i=0; i<size; i++) {
        cout << i << " : ";
        cin >> *(arr+i);
    }

    cout << "\n ### 결과 출력 ###" << endl;
    for(i=0; i<size; i++)
        cout << i << " : " << *(arr+i) << endl;

    delete []arr;
    return 0;
}
```



```
C:\WDOCUMENTS AN...
입력 할 학생 수 : 3

### 데이터 입력 ###
1 : 10
2 : 20
3 : 30

### 결과 출력 ###
1 : 10
2 : 20
3 : 30
Press any key to continue
```

main 함수 안에서
=) 동적 메모리 할당
포인터.
(Heap)

객체지향 프로그래밍

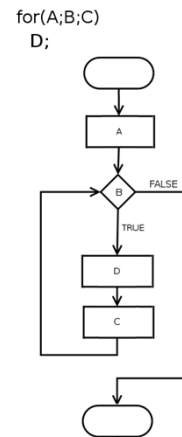


- C++ 프로그래밍 기초
- 객체지향 프로그래밍
 - 클래스와 데이터 추상화
 - 연산자 다중 정의
 - 상속과 다형성
 - C++ 입출력
 - 예외 처리
 - 템플릿과 STL



참고문헌

- [1] 윤성우, "윤성우의 열혈 C++ 프로그래밍"(개정판), 2010.
- [2] H.M. Deitel, P. J. Deitel, "C++ HOW TO PROGRAM: 10th Edition", Prentice Hall, 2017.
- [3] Paul Deitel, Harvey Deitel, "C How to Program", Global Edition, 8/E, Pearson, 2016.
- [4] Michael T. Goodrich 외 2인 지음, 김유성 외 2인 옮김, "C++로 구현하는 자료구조와 알고리즘", 한티에듀, 2020.



이 강의자료는 저작권법에 따라 보호받는 저작물이므로 무단 전제와 무단 복제를 금지하며,
내용의 전부 또는 일부를 이용하려면 반드시 저작권자의 서면 동의를 받아야 합니다.

Copyright © Clickseo.com. All rights reserved.

