



컴퓨터프로그래밍 및 실습

화 3,4 (5313) / 목 1,2 (5402)

RTDCS. 422.

조용아

cyr1212@naver.com

Real-time Operating system for Dependable Embedded Operation
<http://www.rodeolab.org>



컴퓨터 개론 및 실습

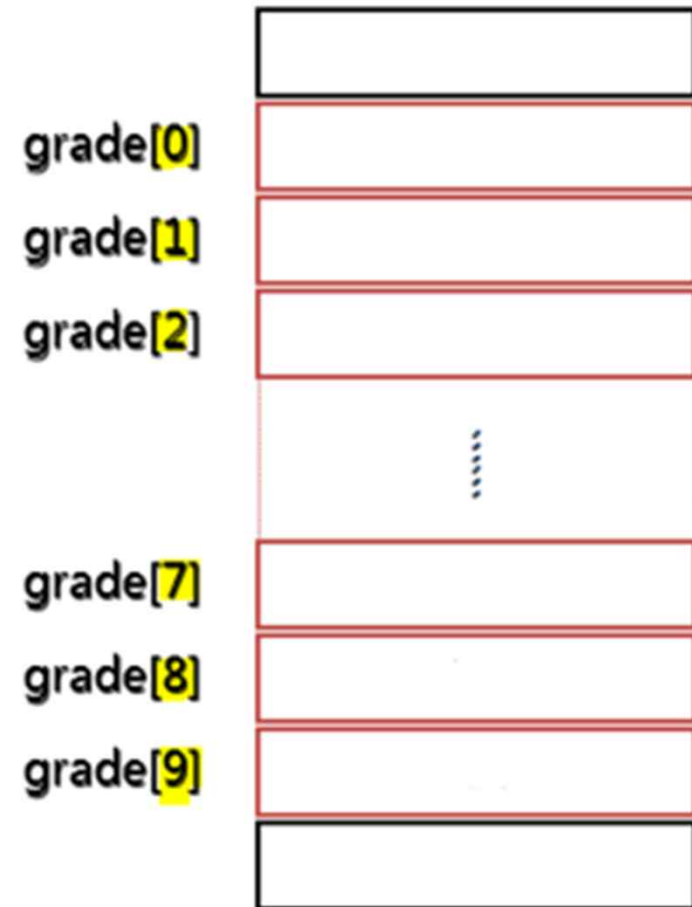
➤ Teaching Assistant

- 조용아 (공대 422호)
- E-mail : cyr1212@naver.com

배열(Array)의 선언

- 자료형 배열이름[배열크기];
- int grade[10];
- 배열 번호는 항상 0부터 시작!

```
int score[60];  
float cost[12];  
char name[50];  
char src[10], dst[10];  
int index, days[7];
```



2차원 배열

- `int s[2][3];`
 행 열

s[0][0]	s[0][1]	s[0][2]
s[1][0]	s[1][1]	s[1][2]

행

s[0][0]	s[0][1]	s[0][2]
s[1][0]	s[1][1]	s[1][2]

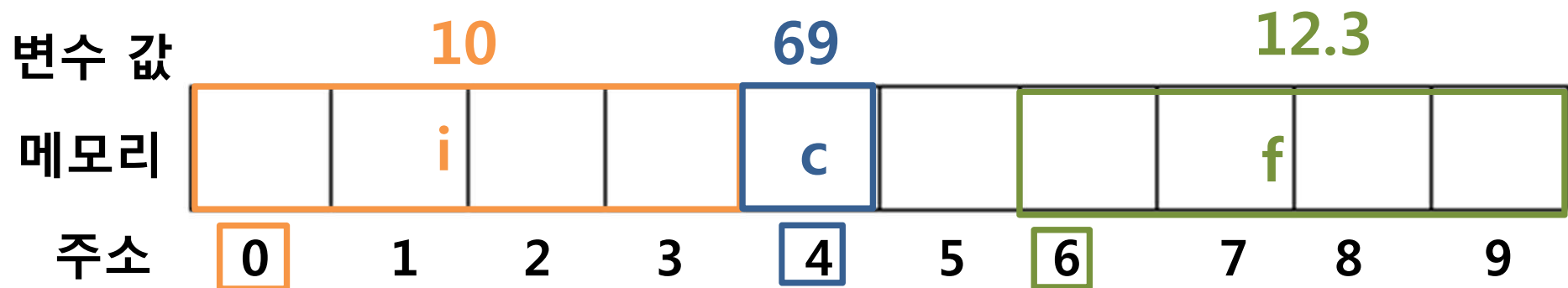
열

포인터(Pointer)

- 포인터(Pointer) : 메모리의 주소를 가지고 있는 변수

변수의 메모리 저장 2

- &(주소 연산자) : 변수의 주소를 계산하는 연산자



```
cout << &i << endl; // -> 0
cout << &c << endl; // -> 4
cout << &f << endl; // -> 6
```

포인터(Pointer)

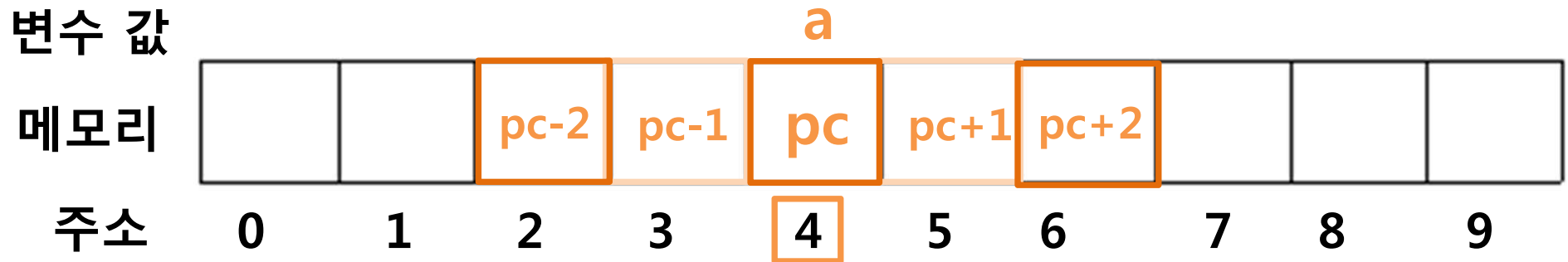
- *(간접 참조 연산자) : 포인터가 가리키는 값을 가져오는 연산자

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int i = 10;
8      int *p = &i;
9
10     cout << *p;
11
12     return 0;
13 }
```

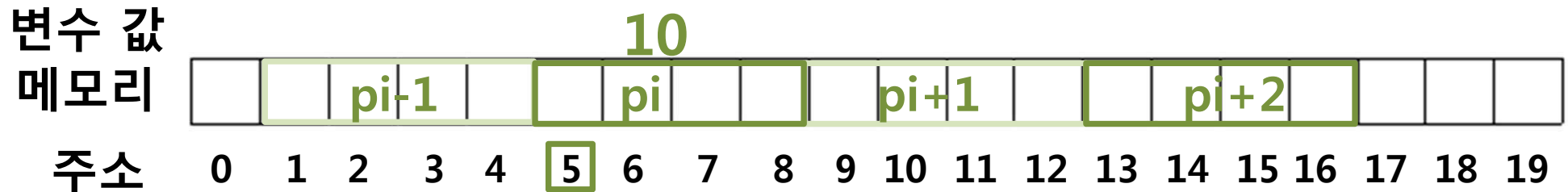
100이 출력

포인터 연산 예제 2

- char형 포인터의 증감



- int형 포인터의 증감



포인터와 배열

- 배열 이름 = 포인터
- 배열 이름은 첫 번째 배열 원소의 주소와 같음

배열과 함수

- 배열 이름 = 포인터
- 배열 이름은 첫 번째 배열 원소의 주소와 같음
- 배열을 함수의 인수로 전달하는 경우 배열 이름이 포인터이므로 배열의 첫 번째 배열 원소의 주소가 전달

배열과 함수 예제

```
1 #include <iostream>
2
3 int get_avg(const int score[], int n);
4 void increment(int score[], int n);
5
6 using namespace std;
7
8 int main()
9 {
10     const int STUDENTS = 5;
11     int grade[STUDENTS] = {1, 2, 3, 4, 5};
12     int avg;
13
14     increment(grade, STUDENTS);
15     avg = get_avg(grade, STUDENTS);
16     cout << "평균 " << avg << endl;
17
18     return 0;
19 }
20
```

```
21 void increment(int score[], int n)
22 {
23     int i;
24
25     for(i=0; i<n; i++)
26         ++score[i];
27 }
28
29 int get_avg(const int score[], int n)
30 {
31     int i;
32     int sum = 0;
33
34     for(i=0; i<n; i++)
35         sum += score[i];
36
37     return sum/n;
38 }
```

메모리 할당 받는 방법(1/2)

- 메모리 할당 받는 방법

1. 정적(static)

- 프로그램이 시작되기 전에 미리 정해진 크기의 메모리를 할당 받는 것
- 프로그램의 실행 도중 그 크기가 변경될 수 없음

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int i, j;
8      int S[100];
9
10     return 0;
11 }
```

메모리 할당 받는 방법(2/2)

2. 동적(dynamic)

- 프로그램 실행 도중에 동적으로 메모리를 할당 받는 것
- 필요한 만큼의 메모리를 시스템으로부터 할당 받아서 사용
- 사용이 끝나면 시스템에 메모리를 반납
- new와 delete 키워드 사용

동적 메모리 할당 과정

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int *pi;
8      int i, num = 1;
9
10     pi = new int[100];
11
12     for(i=0; i<100; i++)
13         *(pi+i) = num++;
14
15     cout << *(pi+0) << endl;
16     cout << *(pi+50) << endl;
17
18     delete[] pi;
19
20     return 0;
21 }
```

동적 메모리 할당

동적 메모리 사용

동적 메모리 반납

동적 메모리 할당, 반납

```
int *pi = new int;  
int *pia = new int[100];  
double *pd = new double;  
double *pda = new double[100];  
  
delete pi;  
delete[] pia;  
delete pd;  
delete[] pda;
```

동적 메모리의 사용이 끝난 후 반납하지 않으면,
메모리 부족으로 프로그램이 중지될 수 있음

참조자

- 참조자(reference) : 변수에 별명을 붙이는 것

```
int var;  
int &ref = var;
```

참조자 ref는 변수 var의 별명(alias)

참조자 예제

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int var;
8      int &ref = var;
9
10     var = 10;
11     cout << "var의 값 = " << var << endl;
12     cout << "ref의 값 = " << var << endl;
13
14     ref = 20;
15     cout << "var의 값 = " << var << endl;
16     cout << "ref의 값 = " << var << endl;
17
18     return 0;
19 }
```

참조자 선언

ref의 값 변경 -> var의 값도 변경

참조자와 포인터

- 참조자는 반드시 선언과 동시에 초기화
- 포인터는 변경가능, 참조자는 변경 불가능
- 참조자를 상수로 초기화하면 컴파일 에러

참조자

- call by reference의 또 다른 형태
- 참조자를 통한 변경을 방지하려면, const를 붙여줌

```
1  #include <iostream>
2
3  void print(const int& r);
4
5  using namespace std;
6
7  int main()
8  {
9      print(100);
10
11     return 0;
12 }
13
```

오류 예방 목적

```
14 void print(const int& r)
15 {
16     cout << "원래의 값 = " << r << endl;
17     return;
18 }
```

참조자와 포인터 예제

```
1 #include <iostream>
2
3 void dec_by_r(int &r);
4 void dec_by_p(int *p);
5
6 using namespace std;
7
8 int main()
9 {
10     int time = 10;
11
12     cout << "1 _ time = " << time << endl;
13
14     dec_by_r(time); time을 전달
15     cout << "2 _ time = " << time << endl;
16
17     dec_by_p(&time); time의 주소 전달
18     cout << "3 _ time = " << time << endl;
19
20     return 0;
21 }
```

r은 time의 참조자.
call by reference(암묵적)

```
22
23 void dec_by_r(int &r)
24 {
25     r--;
26     return;
27 }
```

call by reference(명시적)

```
28
29 void dec_by_p(int *p)
30 {
31     --(*p);
32     return;
33 }
```

포인터와 함수

- C++에서의 인수 전달 방법
 - 값에 의한 호출(call by value)
 - > 함수로 복사본이 전달
 - 참조에 의한 호출(call by reference)
 - > 함수로 주소가 전달,
원본의 변경이 가능

참조에 의한 호출

- C++에서의 인수 전달 방법
 - 값에 의한 호출(call by value)
 - > 함수로 복사본이 전달

- 참조에 의한 호출(call by reference)
 - > 함수로 주소가 전달,
원본의 변경이 가능
- 포인터를 사용하는 방법

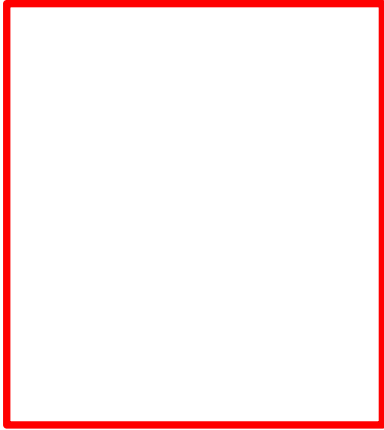
참조자를 사용하는 방법

call by value 예제(1/2)

```
1  #include <iostream>
2
3  void swap(int x, int y);
4
5  using namespace std;
6
7  int main()
8  {
9      int a = 100, b = 200;
10
11     cout << "swap() 호출 전 : a = " << a << " b = " << b << endl;
12
13     swap(a, b);
14
15     cout << "swap() 호출 후 : a = " << a << " b = " << b << endl;
16
17     return 0;
18 }
```

call by value 예제(2/2)

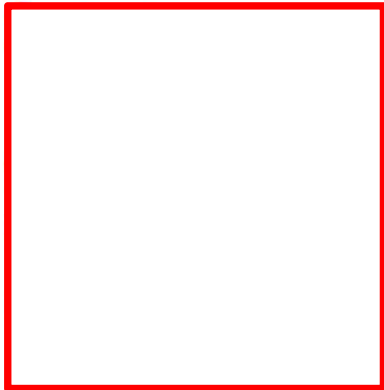
```
19  
20 void swap(int x, int y)  
21 {  
22  
23  
24  
25  
26  
27 }
```



call by reference 예제 – pointer(1/2)

```
1  #include <iostream>
2
3  void swap(int *px, int *py);
4
5  using namespace std;
6
7  int main()
8  {
9      int a = 100, b = 200;
10
11     cout << "swap() 전 : a = " << a << " b = " << b << endl;
12
13     swap(&a, &b);
14
15     cout << "swap() 후 : a = " << a << " b = " << b << endl;
16
17     return 0;
18 }
```

call by reference 예제 – pointer(2/2)

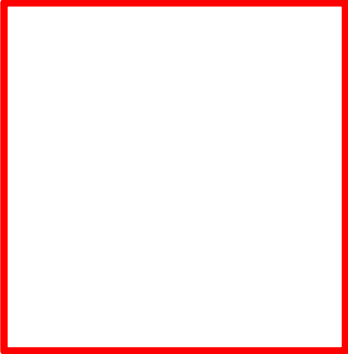
```
19  
20 void swap(int *px, int *py)  
21 {  
22       
23  
24  
25  
26  
27 }
```

call by reference 예제 – reference(1/2)

```
1  #include <iostream>
2
3  void swap(int &rx, int &ry);
4
5  using namespace std;
6
7  int main()
8  {
9      int a = 100, b = 200;
10
11     cout << "swap() 전 : a = " << a << " b = " << b << endl;
12
13     swap(a, b);
14
15     cout << "swap() 후 : a = " << a << " b = " << b << endl;
16
17     return 0;
18 }
```

call by reference 예제 – reference(2/2)

```
19  
20 void swap(int &rx, int &ry)  
21 {  
22  
23  
24  
25  
26  
27 }
```



C++에서 문자열을 나타내는 방법

- C++에서 문자열을 나타내는 방법
 - C - 문자열
 - string 클래스 객체

문자열(string) : 문자들이 여러 개 모인 것
"A"
"Hello World!"

C - 문자열

- 하나의 문자 : char 형 변수로 저장
- 문자열 : char 형 배열로 저장

문자 배열의 초기화

- 문자 배열 원소들을 중괄호 안에 넣어주는 방법

```
char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

- 문자열 상수를 사용하여 초기화하는 방법

```
char str[6] = "Hello";
```

- 배열 크기를 지정하지 않는 경우(선언과 동시에 초기화해야 함)

```
char str[] = "C_string";
```

문자 배열에 문자를 저장

- 문자를 개별적으로 대입

```
str[0] = 'H';  
str[1] = 'e';  
str[2] = 'l';  
str[3] = 'l';  
str[4] = 'o';  
str[5] = '\n';
```

- strcpy()를 사용하여 문자열을 문자 배열에 복사

```
strcpy(str, "Hello");
```


문자열 예제

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      char str1[7] = "Seoul_";
8      char str2[3] = {'i', 's'};
9      char str3[] = "_the capital city of Korea";
10
11     cout << str1 << str2 << str3 << endl;
12
13     return 0;
14 }
```

문자열 처리 라이브러리

함수	설명
strlen(s)	문자열 s의 길이를 구한다.
strcpy(s1, s2)	s2를 s1에 복사한다.
strcat(s1, s2)	s2를 s1의 끝에 붙여넣는다.
strcmp(s1, s2)	s1과 s2를 비교한다.
strncpy(s1, s2, n)	s2의 최대 n개의 문자를 s1에 복사한다.
strncat(s1, s2, n)	s2의 최대 n개의 문자를 s1의 끝에 붙여넣는다.
strncmp(s1, s2, n)	최대 n개의 문자까지 s1과 s2를 비교한다.
strchr(s, c)	문자열 s안에서 문자 c를 찾는다.
strstr(s1, s2)	문자열 s1에서 문자열 s2를 찾는다.

실습 1

- 문자열의 길이 출력 프로그램
- 사용자로부터 문자열을 입력 받음
- 문자열의 길이를 계산하여 출력
(문자열의 길이 계산시 포인터 이용)

실습 2

- 문자의 빈도 출력 프로그램
- 사용자로부터 문자열을 입력 받음
- 각 문자의 빈도를 계산하여 출력
(포인터 사용할 것)

실습 3

- 문자열 공백 구분 출력 프로그램
- 사용자로부터 문자열을 입력 받음
- 문자열을 공백으로 구분하여 출력하는 함수를 작성

```
void splitString(char* str, char delim);
```

```
char *str == char str[]
```

- 위와 같은 함수 원형을 가지는 함수를 작성
- 문자열을 입력 받아 다음과 같이 함수를 호출

```
splitString(str, ' ');
```

실습 4

- Palindrome 판단 프로그램
- Palindrome : 앞으로 읽었을 때와 뒤로 읽었을 때 동일한 숫자 또는 문자열을 뜻함
- 사용자로부터 문자열을 입력 받음
- 입력 받은 문자열이 palindrome인지 판단하는 함수를 작성
- 아래와 같은 함수의 원형을 작성
`int isPalindrome(char* str);`