

기초 학습

C++ 프로그램 구성

2

basic.c

C 컴파일

```
#include <stdio.h>

int g=20; /* 전역 변수 */

int add(int x, int y) { /* 전역 함수 */
    return x + y;
}

int main() {
    int a, b, sum; /* 지역 변수 */
    scanf("%d", &a, &b); /* 입력 */
    sum = a + b;
    printf("%d", sum); /* 출력 */
    return 0;
}
```

2 5
7

키 입력

basic.cpp

C++ 컴파일

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

int g=20; /* 전역 변수 */

int add(int x, int y) { // 전역 함수
    return x + y;
}

int main() {
    int a, b, sum; // 지역 변수
    cin >> a >> b; // 입력
    sum = a + b;
    cout << sum; // 출력
    return 0;
}
```

2 5
7

키 입력

기본적인 구성에 있어서 C/C++ 비교

3

□ 비교

- #include 사용 - 헤더 파일 첨부
- 변수와 변수 선언 - 변수 타입과 선언 방법 동일
- 함수 구성 및 함수 호출 - 함수 작성과 호출 방법 동일
- main() 함수 - 프로그램 실행 시작. main() 함수의 원형 동일
- 연산자 - C++는 C 언어의 연산자를 그대로 수용
- 전역 변수와 지역 변수 - C/C++ 동일

□ 변경 추가

- 주석문 - /* */에 한 줄짜리 주석(//) 추가
- 표준 입출력 헤더 파일 - <stdio.h>에 <iostream> 추가.
 - C++에서는 2003년부터 헤더 파일에 .h 사용하기 않음
- 표준 입출력 방법 - C 표준입출력 함수 scanf/printf에 cin/cout 객체 추가

□ C++ 데이터 타입 - bool 추가

- void, char, int, short int, long, float, double, bool
- bool 타입 추가
 - bool 타입의 상수 true, false 사용 가능

예제 1-1 기본 C++ 프로그램

4

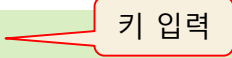
```
#include <iostream>
using namespace std;

int g=20; /* 전역 변수 */

int add(int x, int y) { // 전역 함수
    return x + y; // x와 y의 합 리턴
}

int main() {
    int a, b, sum; // 지역 변수
    cout << "두 정수를 입력하세요 >>"; // 프롬프트 출력
    cin >> a >> b; // 두 정수를 읽어 a와 b에 입력
    sum = a + b;
    cout << "합은 " << sum << "\n"; // sum 값 출력
    cout << "합은 " << add(a, b) << "\n"; // add() 함수 호출 결과 출력
    cout << "전역 변수 g 값은 " << g; // g 값 출력

    return 0; // return 문을 생략하면 자동으로 return 0;이 삽입된다.
}
```

두 정수를 입력하세요 >>33 55  키 입력
합은 88
합은 88
전역 변수 g 값은 20

C/C++ 연산자

5

□ int a = 5, b = 10, c;

(1) a + b/3 * 3

(2) b << 2

(3) a != b

(4) b % a

(5) (a>b)?a:b

(6) sizeof(a)

(7) c = a++; 이후의 c 값

(8) a += b; 이후의 a 값

(9) a & b

(10) c = (a + b, a - b);

조건문

6

- C++는 C 언어의 다음 2가지 유형의 조건문 그대로 사용
 - ▣ if, if-else, if-elseif-else
 - ▣ switch

예제 3-1 if-else 사용

7

점수를 입력 받아 90~100 사이이면 A, 80~89 사이이면 B, 70~79 사이이면 C, 60~69 사이이면 D, 그 이하이면 "F 입니다"를 출력하라. 100보다 크거나 음수가 입력되면 "잘못된 점수입니다"를 출력하라.

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

int main() {
    int score;
    cout << "점수를 입력하세요>>";
    cin >> score;
    if(score > 100 || score < 0) {
        cout << "잘못된 점수 입니다";
        return 0;
    }
    if(score >= 90) // 90이상 100이하
        cout << "A 입니다";
    else if(score >= 80) // 80이상 89이하
        cout << "B 입니다";
    else if(score >= 70) // 70이상 79이하
        cout << "C 입니다";
    else if(score >= 60) // 60이상 69이하
        cout << "D 입니다";
    else // 0이상 59이하
        cout << "F 입니다";
}
```

점수를 입력하세요>>85
B 입니다

예제 3-2 switch 사용 예제

8

예제 3-1의 if-else를 switch를 이용하여 구현하라.
점수를 입력 받고 10으로 나눈 몫으로 switch 문을 만들면 된다.

점수를 입력하세요>>85
B 입니다

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

int main() {
    int score, div;
    cout << "점수를 입력하세요>>";
    cin >> score;
    if(score > 100 || score < 0) {
        cout << "잘못된 점수 입니다";
        return 0;
    }
    div = score/10;
    switch(div) {
        case 10:
            case 9:
                cout << "A 입니다"; break; // 90~100 경우
            case 8:
                cout << "B 입니다"; break; // 80 점대
            case 7:
                cout << "C 입니다"; break; // 70 점대
            case 6:
                cout << "D 입니다"; break; // 60 점대
            default : // 나머지 점수 대
                cout << "F 입니다"; break;
    }
}
```

case 문은 break 만나야 빠져나옴

반복문

9

- C++는 C 언어의 다음 3가지 반복문을 그대로 사용
 - ▣ for, while, do-while
 - ▣ 추가된 것 없음
- break
 - ▣ 반복문을 벗어남
- continue
 - ▣ 다음 반복 실행

예제 4-1 for 문 예제

10

두 정수 a, b를 입력 받아 a에서 b까지의 정수 합을 출력하라. 반드시 작은 수, 큰 수 순서로 입력하라.

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

int main() {
    int i, a, b, sum=0;
    cout << "두 개의 정수 입력>>";
    cin >> a >> b;

    for(i=a; i<=b; i++) { // a에서 b까지 합 계산
        sum += i;
    }

    cout << a << "에서 " << b << "까지 합은 " << sum;
}
```

두 개의 정수 입력>>3 6
3에서 6까지 합은 18

예제 4-2 while 문 예제

11

예제 4-1의 for 문을 while 문으로 바꾸어 작성하라.

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

int main() {
    int i, a, b, sum=0;
    cout << "두 개의 정수 입력>>";
    cin >> a >> b;

    i=a;
    while(i<=b) { // i가 b보다 작거나 같은 동안 반복
        sum += i;
        i++;
    }

    cout << a << "에서 " << b << "까지 합은 " << sum;
}
```

두 개의 정수 입력>>3 6
3에서 6까지 합은 18

예제 4-3 do-while 문 예제

12

예제 4-1의 for 문을 do-while 문으로 바꾸어 작성하라.

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

int main() {
    int i, a, b, sum=0;
    cout << "두 개의 정수 입력>>";
    cin >> a >> b;

    i=a;
    do {
        sum += i;
        i++;
    } while(i<=b); // i가 b보다 작거나 같은 동안 반복

    cout << a << "에서 " << b << "까지 합은 " << sum;
}
```

두 개의 정수 입력>>3 6
3에서 6까지 합은 18

예제 4-4 continue와 break 문

13

while 문과 continue, break 문을 이용하여, 정수를 입력 받고 3의 배수이면 "Yes"를, 아니면 "No"를 출력하라. 0이 입력되면 프로그램을 종료한다.

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

int main() {
    int a;
    while(true) {
        cout << "정수 입력>>";
        cin >> a;
        if(a == 0)
            break; // 0이 입력되면 while 문을 벗어남
        if(a%3 != 0) {
            cout << "No" << "\n";
            continue; // 다음 반복. while 문으로 분기
        }
        cout << "Yes" << "\n"; // 입력된 3의 배수 출력
    }
}
```

```
정수 입력>>381
Yes
정수 입력>>297
Yes
정수 입력>>235
No
정수 입력>>0
```

□ 배열이란

- ▣ 동일 타입의 데이터를 하나의 단위로 다루기 위해 연결된 메모리
- ▣ C++는 C 언어의 배열 그대로 사용

□ 배열 선언

```
int n[10]; // 정수 10개짜리 빈 메모리 공간
double d[] = {0.1, 0.2, 0.5, 3.9}; // d의 크기는 자동으로 3으로 설정.
// 배열 d에 순서대로 0.1, 0.2, 0.5, 3.9로 초기화
```

| | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|
| | n[0] | n[1] | n[2] | n[3] | n[4] | n[5] | n[6] | n[7] | n[8] | n[9] |
| 배열 n | | | | | | | | | | |

| | | | | |
|------|------|------|------|------|
| | d[0] | d[1] | d[2] | d[3] |
| 배열 d | 0.1 | 0.2 | 0.5 | 3.9 |

```
n[10] = 20; // 인덱스 10 사용 오류. 인덱스는 0~9까지만 사용 가능
d[-1] = 9.9; // 인덱스 -1 사용 오류. 인덱스로 음수 사용 불가
```

2차원 배열

15

```
int m[2][5]; // 2행 5열의 2차원 배열 선언
```

배열 m

| | | | | |
|---------|---------|---------|---------|---------|
| m[0][0] | m[0][1] | m[0][2] | m[0][3] | m[0][4] |
| | | | | |
| m[1][0] | m[1][1] | m[1][2] | m[1][3] | m[1][4] |
| | | | | |

m[2][0] = 5; // 오류. 인덱스 2가 잘못 사용되었음. 0~1만 가능
m[0][6] = 2; // 오류. 인덱스 6이 잘못 사용되었음. 0~4만 가능

예제 5-1 배열 선언 및 활용

16

배열을 선언하고 배열을 활용하는 코드를 사례를 보인다.

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

int main() {
    int n[10]; // 정수 10개짜리 빈 메모리 공간
    double d[] = {0.1, 0.2, 0.5, 3.9}; // 배열 d에 0.1, 0.2, 0.5, 3.9로 초기화

    int i;
    for(i=0; i<10; i++) n[i] = i*2; // 2의 배수로 n에 값을 채움
    for(i=0; i<10; i++) cout << n[i] << ' '; // 배열 n 출력
    cout << "\n"; // 한 줄 띄다.

    double sum = 0; // C++에서는 필요할 때 변수를 아무 곳이나 선언 가능
    for(i=0; i<4; i++) { // 배열 d의 합 계산
        sum += d[i];
    }

    cout << "배열 d의 합은 " << sum; // 배열 d의 합 출력
}
```

0 2 4 6 8 10 12 14 16 18
배열 d의 합은 4.7

□ 함수란

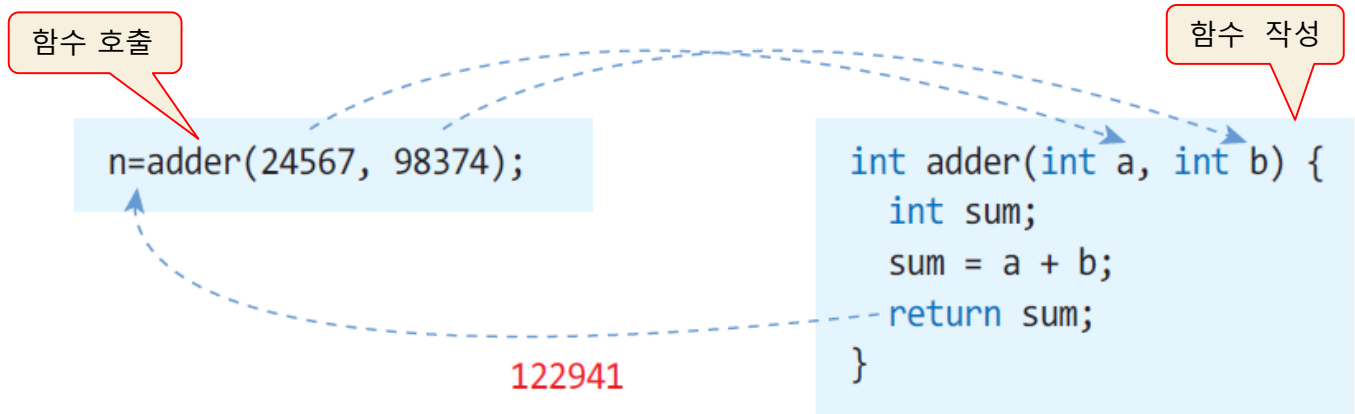
- ▣ 매개변수를 통해 데이터를 전달받아 처리한 후 결과를 리턴하는 코드 블록
- ▣ C++는 C 언어의 함수 기법 그대로 계승



함수의 구성과 함수 호출

18

```
리턴타입 함수이름(매개변수 리스트) {  
    계산하는 프로그램 코드들  
    결과를 리턴하는 return 문  
}
```



예제 6-1 adder() 함수와 호출

19

두 개의 정수를 전달받아 합을 리턴하는 함수 adder()를 작성하라.

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

// 두 개의 정수를 받아 합을 구하고 결과를 리턴하는 함수 adder
int adder(int a, int b) {
    int sum;
    sum = a + b;
    return sum;
}

int main() {
    int n = adder(24567, 98374); // 함수 adder() 호출
    cout << "24567과 98374의 합은 " << n << "입니다\n";

    int a, b;
    cout << "두 개의 정수를 입력하세요>>";
    cin >> a >> b;
    n = adder(a, b); // 함수 adder() 호출
    cout << a << "와 " << b << "의 합은 " << n << "입니다\n";
}
```

24567과 98374의 합은 122941입니다
두 개의 정수를 입력하세요>>2342 158619
2342와 158619의 합은 160961입니다

함수 작성과 호출시 주의할 점

20

```
void adder2(int a, int b) {  
    int sum;  
    sum = a + b;  
    cout << "합은 " << sum << "입니다";  
}  
...  
adder2(5, 4); // 함수 adder2() 호출
```

호출한 코드에게 아무 값도 리턴
하지 않기 때문에, 리턴 타입 void로 선언

~~int n~~ = adder2(100, 200); // 오류. adder2()는 아무 값도 리턴하지 않음

char c = adder(100, 200); // 경고 혹은 오류. adder()는 정수(int) 값 리턴

adder2(); // 오류. 아무 값도 전달하지 않기 때문
adder2(10); // 오류. 하나의 값(10)만 전달하기 때문
adder2(10, 20, 30); // 오류. 3개의 값을 전달하기 때문

adder2(2.3, 5.5); // 매개 변수 값 왜곡. adder2의 매개 변수 a에는 정수 2, b에는 정수 5가 전달됨

예제 6-2 함수 호출

21

다음 코드에는 2개의 함수가 선언되어 있다. 빈칸에 함수를 호출하는 문을 작성하라.

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

// 두 개의 정수를 받아 큰 값을 리턴하는 함수
int bigger(int a, int b) {
    if(a>b) return a;
    else return b;
}

// 매개 변수가 3으로 나누어지면 true, 아니면 false를 리턴하는 함수
bool dividedBy3(int n) {
    if(n%3 == 0) return true;
    else return false;
}

int main() {
    int a, b, n;
    cout << "두 개의 정수 입력>>";
    cin >> a >> b;

    _____ // (1) 함수 bigger() 호출
    cout << a << "중 " << b << "중 큰 값은 " << n << "입니다.\n";

    _____ // (2) n이 3의 배수이면
    cout << n << "은 " << "3의 배수입니다.\n";
    else
        cout << n << "은 " << "3의 배수가 아닙니다.\n";
}
```

두 개의 정수 입력>>23 56
23중 56중 큰 값은 56입니다.
56은 3의 배수가 아닙니다.

(1) n = bigger(a, b);
(2) if(dividedBy3(n))

함수 원형, 함수 프로토타입

22

- 함수 원형, 함수 프로토타입(prototype)이란?
 - ▣ 변수 선언처럼, 함수의 형식만 선언한 것
 - ▣ 세미콜론(;)으로 끝맺음
 - ▣ 예) ***int adder(int a, int b);*** // *adder()* 함수의 원형

- 함수의 원형을 선언하는 이유
 - ▣ 함수 이름, 매개 변수 타입과 개수, 리턴 타입을 컴파일러에게 알려주어 함수 호출 문장이 정확한지 판단하게 도움

함수 원형이 선언된 경우와 아닌 경우

23

adder 이름을 발견할 수 없어 컴파일 오류

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

```
int main() {
    int a, b, sum;
    cin >> a >> b;
    sum = adder(a, b); // 함수 호출 오류
    cout << sum;
    return 0;
}

int adder(int x, int y) {
    return x + y;
}
```

컴파일 오류

(b) adder() 함수 원형을 선언하지 않아
adder() 호출 시 오류 발생

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

```
int adder(int x, int y); // 함수 원형 선언
```

```
int main() {
    int a, b, sum;
    cin >> a >> b;
    sum = adder(a, b); // 함수 호출
    cout << sum;
    return 0;
}

int adder(int x, int y) {
    return x + y;
}
```

adder()
원형 발견

(b) adder() 함수 원형을 선언하여
정상 컴파일된 경우

매개 변수로 배열 전달

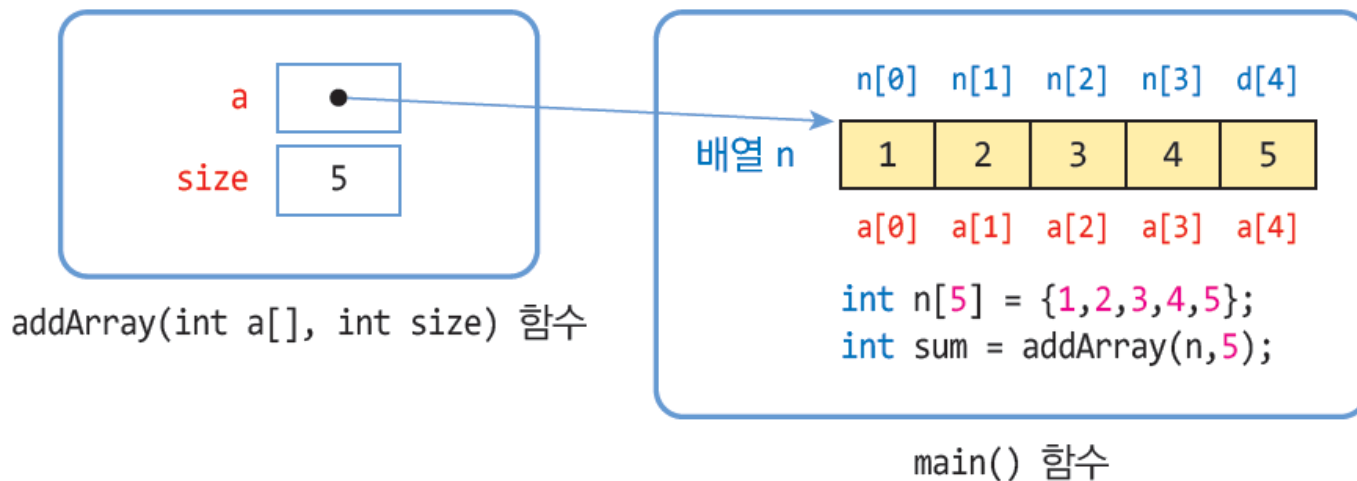
24

□ 함수의 매개 변수로 배열 전달

```
int addArray(int a[], int size) { // 배열을 매개 변수로 가진 함수
...
}
```

```
int n[5] = { 1,2,3,4,5 };
int s = addArray(n, 5); // 배열 n을 매개 변수로 전달
int m[3] = { 1,2,3 };
int t = addArray(m, 3); // 배열 m을 매개 변수로 전달
```

addArray() 호출



예제 6-3 배열을 매개 변수로 가진 함수의 호출

25

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

int addArray(int a[], int size); // 함수의 원형 선언
void makeDouble(int a[], int size); // 함수의 원형 선언
void printArray(int a[], int size); // 함수의 원형 선언

int main() {
    int n[] = { 1,2,3,4,5 };

    // 배열 n[]과 개수를 매개 변수에 전달
    int sum = addArray(n, 5);
    cout << "배열 n의 합은 " << sum << "입니다\n";

    makeDouble(n, 5); // 배열 n과 개수 5를 매개 변수에 전달
    printArray(n, 5); // 배열 n과 개수 5를 매개 변수에 전달
}
```

```
// 배열과 개수를 전달받아 합을 리턴하는 함수
int addArray(int a[], int size) {
    int i, sum=0;
    for(i=0; i<size; i++)
        sum += a[i];
    return sum;
}

// 배열의 값을 두 배로 증가시키는 함수
void makeDouble(int a[], int size) {
    int i;
    for(i=0; i<size; i++)
        a[i] *= 2; // 원소의 값을 2배 증가
}

// 배열을 출력하는 함수
void printArray(int a[], int size) {
    int i;
    for(i=0; i<size; i++)
        cout << a[i] << ' '; // 원소 출력
    cout << "\n";
}
```

배열 n의 합은 15입니다
2 4 6 8 10

포인터

26

□ 포인터란?

- ▣ 포인터(pointer)는 실행 중 메모리의 주소 값
- ▣ 주소(포인터)를 이용하여 메모리에 직접 값을 쓰거나 메모리로부터 값을 읽어올 수 있음

□ 변수와 메모리 주소

```
int n;  
n = 3;
```

- ▣ 변수 n은 정수를 저장할 메모리 공간에 대한 이름, 이곳에 3 기록
- ▣ 값 3이 메모리 몇 번지에 기록되는지 알 수 없음
 - 프로그램이 실행을 시작할 때, 변수 n의 절대 메모리 주소가 정해짐
- ▣ 주소를 사용하는 것보다 이름 n을 사용하는 것이 용이함

포인터 변수 선언

27

□ 포인터 변수

▣ 포인터 즉 주소를 저장하는 변수

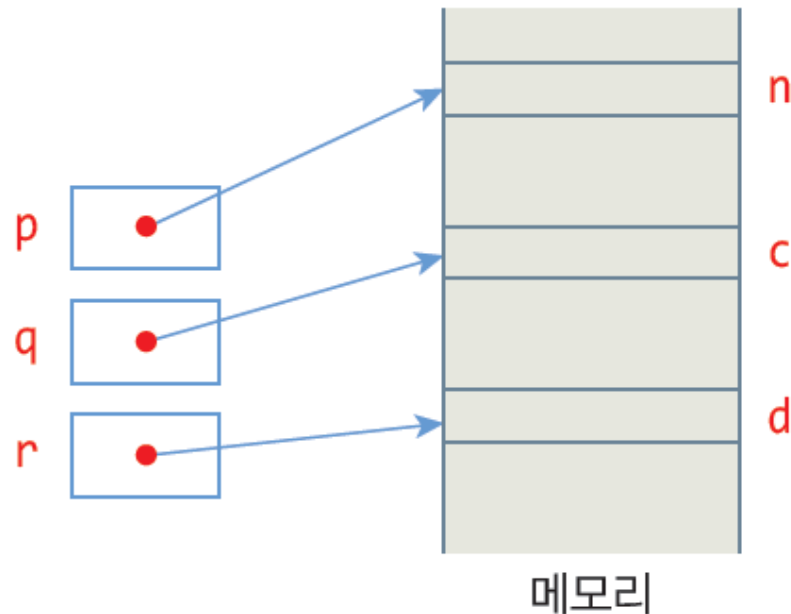
```
int *p; // 정수를 저장하는 메모리에 대한 포인터 변수 p 선언  
p = &n; // p에 n의 주소를 저장
```

```
int n;  
char c;  
double d;
```

```
int *p = &n;
```

```
char *q = &c;
```

```
double *r = &d;
```



예제 7-1 포인터 선언 및 활용

28

포인터를 이용하여 변수에 들어 있는 값을 출력하는 코드를 보인다.

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

```
int main() {
    int n=10, m;
    char c='A';
    double d;
```

```
    int *p= &n; // p는 n의 주소값을 가짐
    char *q = &c; // q는 c의 주소값을 가짐
    double *r = &d; // r은 d의 주소값을 가짐
```

```
    *p = 25; // n에 25가 저장됨
    *q = 'A'; // c에 문자 'A'가 저장됨
    *r = 3.14; // d에 3.14가 저장됨
    m = *p + 10; // p가 가리키는 값(n 변수값)+10을 m에 저장
```

```
    cout << n << ' ' << *p << "\n"; // 둘 다 25가 출력됨
    cout << c << ' ' << *q << "\n"; // 둘 다 'A'가 출력됨
    cout << d << ' ' << *r << "\n"; // 둘 다 3.14가 출력됨
    cout << m << "\n"; // m 값 35 출력
```

```
}
```

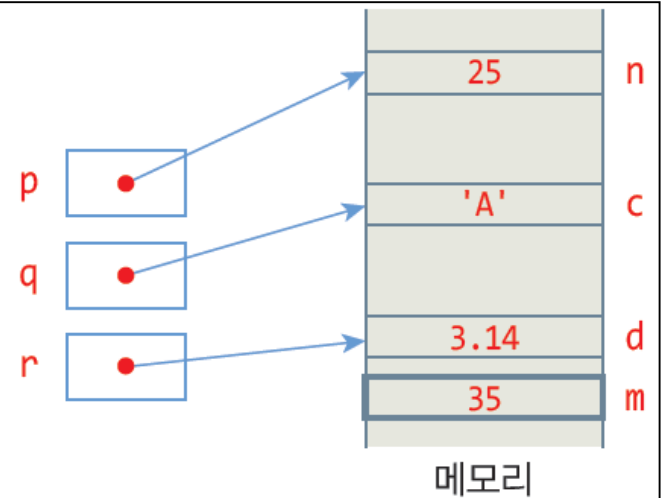
```
int m;

*p = 25;

*q = 'A';

*r = 3.14;

m = *p + 10;
```

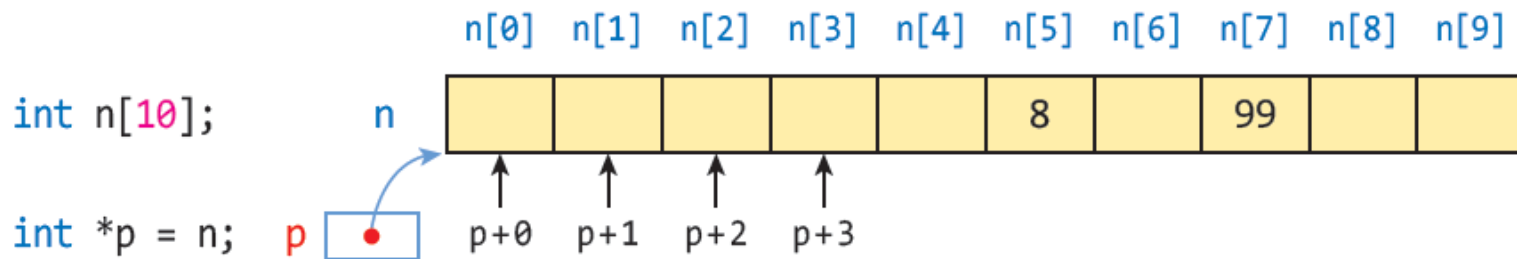


```
25 25
A A
3.14 3.14
35
```

배열과 포인터

29

- 배열 이름은 배열 메모리의 시작 주소로 다름



| | |
|---------------|----------------------------------|
| n[5] = 8 | → 배열 n의 시작 위치에서 5만큼 떨어진 주소에 8 기록 |
| n + 5 | → n[5]의 주소 |
| *(n + 5) = 8; | → n[5]에 8기록 |
| p = p + 7; | → p는 n[7]의 주소 |
| *p = 99; | → n[7]에 99 기록 |

예제 7-2 포인터로 배열 접근

30

배열 n을
3의 배수로 초기화

포인터 p를 이용
하여 배열 n 출력

포인터 p를 이용
하여 배열 n의 원
소 값 2 증가

배열 n 출력

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

int main() {
    int n[10];
    int i;
    int *p;

    for(i=0; i<10; i++)
        *(n+i) = i*3; // 배열의 이름 n을 주소처럼 사용 가능. 배열 n을 3의 배수로 채움

    p = n; // 포인터 p에 배열 n의 시작 주소를 설정한다.
    for(i=0; i<10; i++) {
        cout << *(p+i) << ' '; // 포인터 p를 이용하여 배열 n의 원소 접근
    }
    cout << "\n";

    for(i=0; i<10; i++) {
        *p = *p + 2; // 포인터 p를 이용하여 배열의 원소 값을 2 증가
        p++; // p는 다음 원소의 주소로 증가
    }

    for(i=0; i<10; i++)
        cout << n[i] << ' ';
    cout << "\n";
}
```

0 3 6 9 12 15 18 21 24 27
2 5 8 11 14 17 20 23 26 29

예제 7-3 포인터를 매개 변수로 전달받는 함수

31

포인터로 정수 2개를 전달받아 비교하는 함수 equal()을 작성하라.

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

bool equal(int* p, int* q); // 함수의 원형 선언

int main() {
    int a=5, b=6;
    if(equal(&a, &b)) cout << "equal" << "\n";
    else cout << "not equal" << "\n";
}

bool equal(int* p, int* q) { // 포인터 매개 변수
    if(*p == *q) return true;
    else return false;
}
```

not equal

예제 7-4 배열을 비교하는 함수

32

배열을 비교하는 `equalArray()` 함수를 다음과 같이 작성할 수 있다. `equalArray()` 함수의 매개 변수를 포인터 타입으로 선언하여 재작성하라.

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

bool equalArray(int* p, int* q, int size); // 함수의 원형 선언

int main() {
    int a[] = {1,2,3,4,5};
    int b[] = {1,2,3,4,5};

    if(equalArray(a, b, 5)) cout << "arrays equal" << "\n";
    else cout << "arrays not equal" << "\n";
}

bool equalArray(int* p, int* q, int size) {
    int i;
    for(i=0; i<size; i++) {
        if(*p != *q) return false;
        p++; // p는 배열의 다음 원소를 가리킴
        q++; // q도 배열의 다음 원소를 가리킴
    }
    return true;
}
```

arrays equal