5강. 배열, 포인터, 참조

C++ 프로그래밍

jhhwang@kumoh.ac.kr

컴퓨터공학과 황준하

목차

배열

- 배열
  - 같은 타입의 변수 여러 개를 하나의 변수명으로 처리

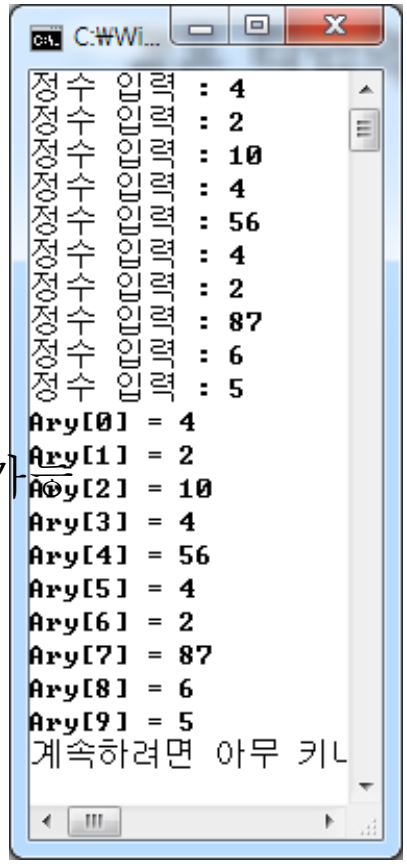
```
void main(void)
{
    int Ary[10];

    for (int i = 0; i < 10; i++) {
        cout << "정수 입력 : ";
        cin >> Ary[i];
    }

    for (int i = 0; i < 10; i++)
        cout << "Ary[" << i << "] = " << Ary[i] << endl;
}
```

총 10개의 변수 : Ary[0]~Ary[9]

변수 인덱스를 통해 해당 원소 접근 가능



다차원 배열과 배열 선언 시 초기화 방법

Ary[0][0]	Ary[0][1]	Ary[0][2]	Ary[0][3]
Ary[1][0]	Ary[1][1]	Ary[1][2]	Ary[1][3]
Ary[2][0]	Ary[2][1]	Ary[2][2]	Ary[2][3]

- 2차원 배열, 3차원 배열, ...
  - int Ary[3][4];
    - Ary[0][0] ~ Ary[2][3] 의 총 12개의 int형 변수가 생김
    - Ary[1][2]와 같이 각 원소에 접근 가능
  - 다차원 배열의 사용 방법 동일
- 배열 선언 및 초기화

```
int Ary1[5] = { 1, 2, 3, 4, 5 };
int Ary2[5] = { 1, 2 };
int Ary3[2][3] = { { 1, 2, 3 }, { 4, 5, 6 } };
int Ary4[2][3] = { { 1 }, { 4, 5 } };
int Ary5[2][3] = { 1, 2 };
```

나머지는 0으로 자동 초기화

C++ 메모리 구조

- 하나의 프로그램 실행(프로세스) 시 생성되는 메모리 구조


```
int a = 1;

int Sum(int x, int y)
{
	static int c;
	c = x + y;
	return c;
}

void main(void)
{
	int b = 2;
	int *p = new int(3);

	cout << Sum(b, *p) << endl;

	delete p;
}
```

결론 : 변수와 함수 모두 메모리에 올라감

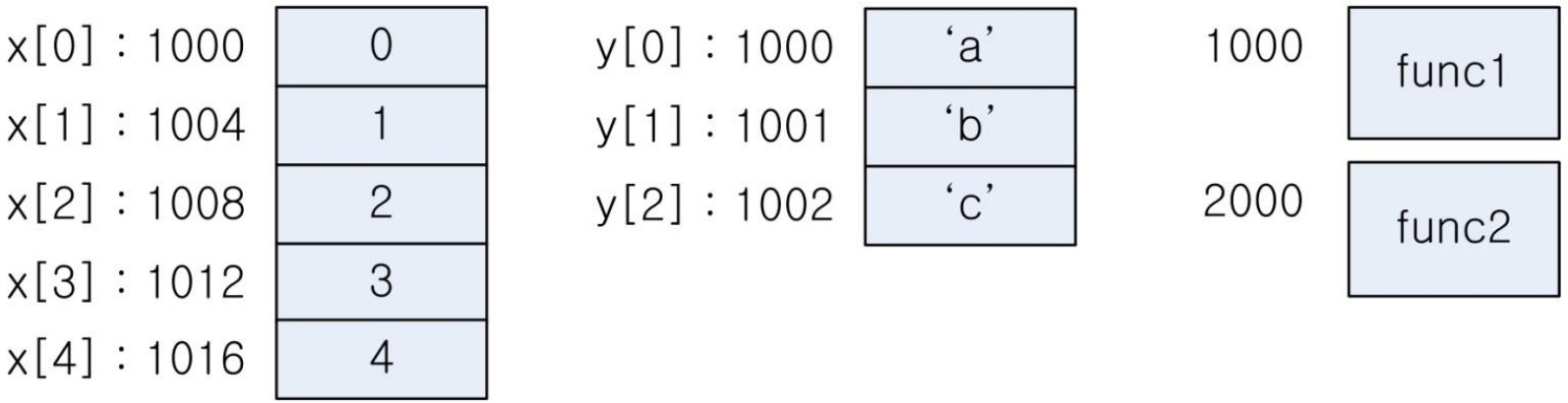
C++ 메모리 구조

- 메모리 주소
  - 주소 : 바이트 단위의 번호, 4바이트로 표현 – 0번지 ~ 2^{32}-1번지
- 변수, 함수는 메모리를 차지함
  - 배열은 연속적인 메모리 공간을 차지함

```
int x[5] =
  { 0, 1, 2, 3, 4 };
```

```
char y[3] =
  { 'a', 'b', 'c' };
```

```
int func1();
int func2();
```



주소 연산자

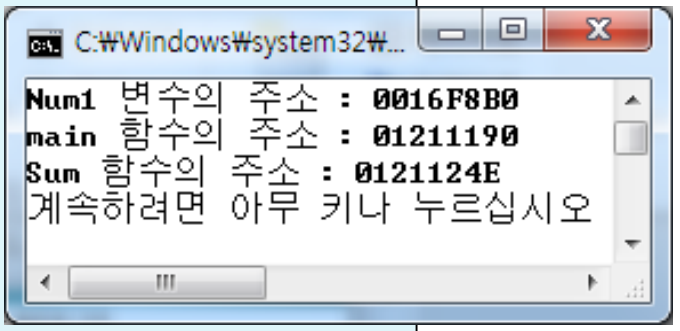
- 변수와 함수가 올라가 있는 메모리 주소값 알아내기
  - 변수 : 주소 연산자(&) 사용

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

void main(void)
{
	int Num1 = 3;

	cout << "Num1 변수의 주소 : " << &Num1 << endl;
	cout << "main 함수의 주소 : " << main << endl;
	cout << "Sum 함수의 주소 : " << Sum << endl;
}
```

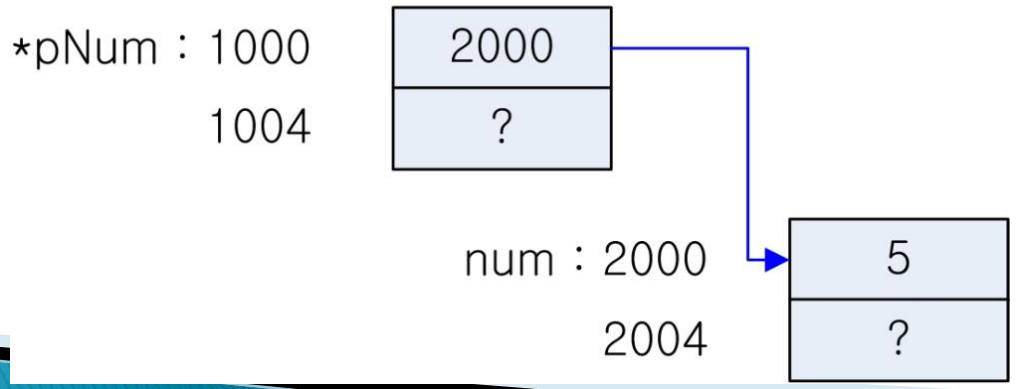
Num1 변수의 주소



포인터

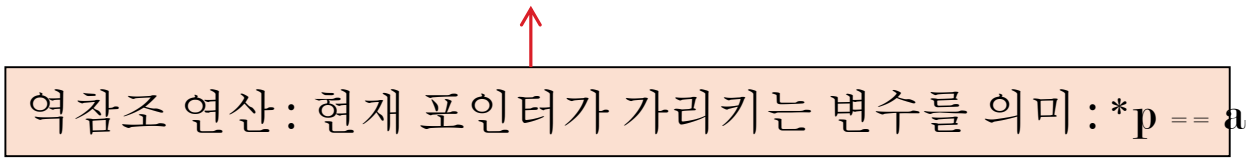
- 포인터 (변수)
  - 주소값을 저장하는 변수
  - 주소값에도 타입이 있음
    - int a; &a → int형 주소값
    - double b; &b → double형 주소값
  - int형 포인터 == int형 주소값을 저장하는 변수
    - int *p;
  - 포인터도 변수다!

```
int num = 5;
int *pNum  = &num;
```



포인터 연산

- 역참조 연산 (*)
  - int a = 3; int *p = &a; *p = 5;



- 포인터 연산
  - 덧셈, 뺄셈만 가능 : +, -, ++, --
  - int *p = &a; p = p + 1;
    - int 타입의 크기인 4만큼 증가 → 다음 int 변수를 가리키는 모양

연습 문제

- 다음 프로그램의 출력 결과는?
  - Num1의 주소는 1000번지, pNum의 주소는 2000번지로 가정

```
int Num1 = 3;
int *pNum = &Num1;

cout << Num1 << endl;
cout << &Num1 << endl;
cout << pNum << endl;
cout << &pNum << endl;

*pNum = 5;
pNum++;

cout << Num1 << endl;
cout << &Num1 << endl;
cout << pNum << endl;
cout << &pNum << endl;
```

실행 결과

```
3
1000
1000
2000
5
1000
1004
2000
```

배열과 포인터

- 배열 이름
  - 첫 번째 원소의 주소를 의미하는 상수
  - **int Ary[5]; Ary == &Ary[0]**

```
void main(void)
{
      int Ary[5]= { 1, 2, 3, 4, 5 };
      int *p = Ary;

      for (int i = 0; i < 5; i++) {
            cout << *p << endl;
            p++;
      }
}
```

첫 번째 원소를 가리킴

다음 원소를 가리킴

배열과 포인터

- 배열은 포인터처럼, 포인터는 배열처럼 사용 가능!
  - 단, 배열명은 상수이므로 변경 불가

```
void main(void)
{
	int Ary[5]= { 1, 2, 3, 4, 5 };
	int *p = Ary;

	for (int i = 0; i < 5; i++) {
		cout << p[i] << " ";
		cout << *(Ary + i) << endl;
	}
}
```

포인터를 배열처럼 사용
p[i] == *(p + i)

배열을 포인터처럼 사용
*(Ary + i) == Ary[i]

배열과 포인터

Ary[0]	Ary[0][0]	Ary[0][1]	Ary[0][2]
Ary[1]	Ary[1][0]	Ary[1][1]	Ary[1][2]

- 다차원 배열명의 의미
  - 첫 번째 원소의 주소!
  - int Ary[2][3]; Ary == &Ary[0], int (*p)[3] = Ary;
    - p++ → 12만큼 증가 (int[3] 단위)
    - P[0][2]와 같이 2차원 배열처럼 사용 가능
  - int Ary[2][3][4]; Ary == &Ary[0], int (*p)[3][4] = Ary;
- 복잡한 포인터
  - int **p; 포인터에 대한 포인터
  - int *p[3]; int 포인터 3개를 원소로 배열
  - int (*p)[3]; int[3] 배열 단위로 가리키는 포인터
    - 해석요령 : pointer of array[3] of int → int 형 1차원 배열(3)에 대한 포인터

연습 문제

- int Ary[3][4] 가 있다. Ary[2][3] = 5;와 동일한 의미가 되도록 Ary를 포인터 방식으로 사용해 보라.

```
*(*(Ary + 2) + 3) = 5;
*(Ary[2] + 3) = 5;
(*(Ary + 2))[3] = 5;
```

- 다음 각 변수가 무엇을 뜻하는지 말해 보라.
  - int **a[3];
  - int (*b)[3][4];
  - int *(*c)[3];

int 포인터의 포인터를 원소로 갖는 1차원 배열[3]

2차원 배열[3][4]에 대한 포인터

int 포인터를 원소 갖는 1차원 배열[3]에 대한 포인터

메모리 동적 할당

- 메모리 동적 할당
  - 메모리가 필요할 때 변수를 사용하지 않고 동적으로 할당 받음
  - 포인터 변수를 통해 가리키고 사용함
- new(메모리 할당)와 delete(해제)

```
void main(void)
{
        int *p;

        p = new int;
        *p = 5;
        cout << *p << endl;
        delete p;
}
```

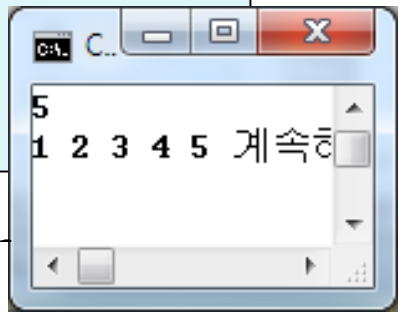
int 1개(4바이트) 할당

동적 메모리 해제

```
        p = new int[5];
        for (int i = 0; i < 5; i++) {
                p[i] = i + 1;
                cout << p[i] << " ";
        }
        delete [] p;
}
```

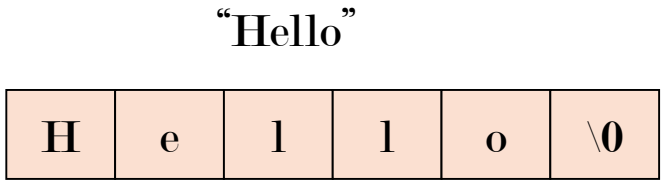
int 5개(20바이트) 할당 배열

배열로 할당받은 경우 반드시 [] 사용



문자열

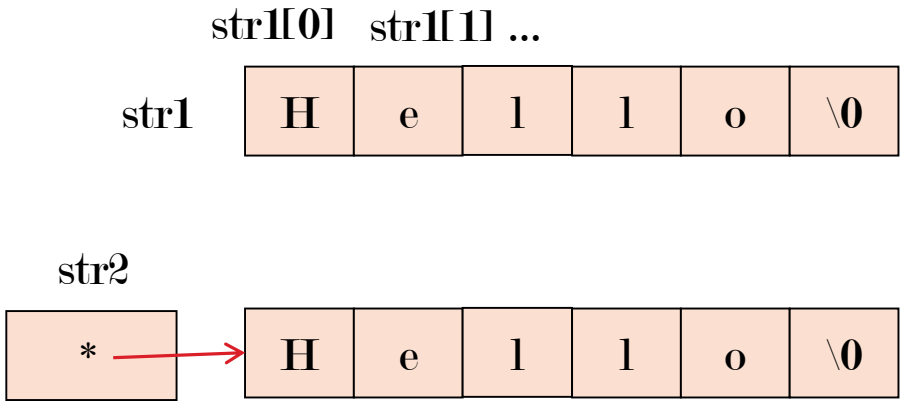
- 문자열 표현 방식
  - char 문자의 집합
  - 단, 마지막 문자로 널문자('\0') 포함



- 문자열의 저장
  - 문자열 상수 : char * 로 가리킬 수 있음, 불변
  - 문자열 변수 : 1차원 배열

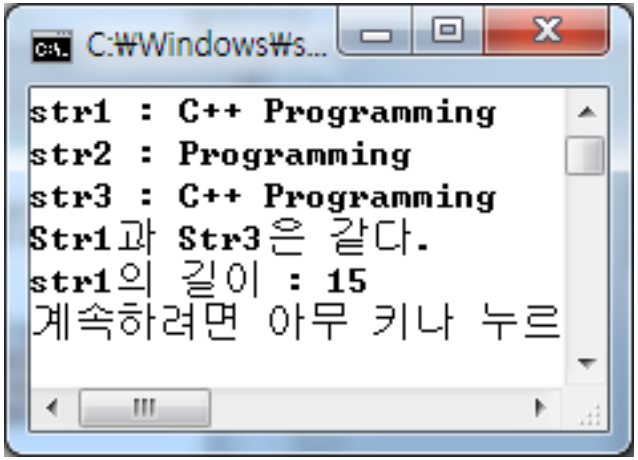
```
void main(void)
{
    char str1[6] = "Hello";
    char *str2 = "Hello";

    cout << str1 << endl;
    cout << str2 << endl;
}
```



문자열 처리

- 문자열 변수에 대한 조작 함수 사용
  - 헤더 파일 : ‹cstring›
  - strlen : 문자열의 길이
  - strcpy : 문자열 복사
  - strcat : 문자열 뒤에 추가
  - strcmp : 문자열 비교



같으면 0, str1이 크면 양수
Str3이 크면 음수 반환

```
void main(void)
{
	char str1[255] = "C++ ";
	char str2[255] = "Programming";
	char str3[255];

	strcat(str1, str2);
	strcpy(str3, str1);

	cout << "str1 : " << str1 << endl;
	cout << "str2 : " << str2 << endl;
	cout << "str3 : " << str3 << endl;

	if (strcmp(str1, str3) == 0)
		cout << "Str1과 Str3은 같다." << endl;

	cout << "str1의 길이 : " << strlen(str1) << end
}
```

참조

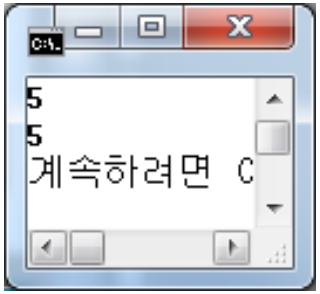
- 참조 변수
  - 기존 변수(변수 하나)의 또 다른 이름
  - 혼자 존재할 수 없음 → 선언과 동시에 초기화 필수
  - 초기화 이후에는 일반 변수와 100% 동일하게 사용!

```
void main(void)
{
	int Num1 = 3;
	int &Num2 = Num1;

	Num2 = 5;

	cout << Num1 << endl;
	cout << Num2 << endl;
}
```

참조 변수 Num2는
Num1과 동일
→ 이후 Num2는 일반변수와
동일함 (사용 방법 등)



참조

- 참조 변수의 예

a, b, c 모두 동일

포인터 변수 p, q 동일

배열 ary와 rAray 동일

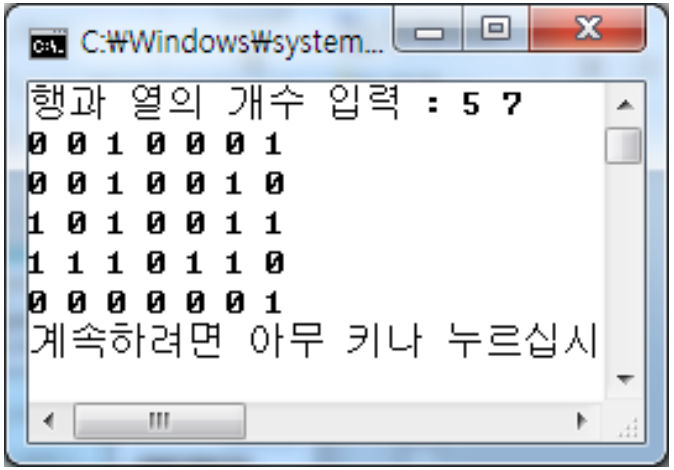
```
void main(void)
{
    int a;
    int &b = a;
    int &c = b;
    c = 5;

    int *p = &a;
    int *&q = p;
    *q = 7;

    int ary[3];
    int (&rAry)[3] = ary;
    rAry[1] = 3;
}
```

프로그래밍 연습

- 사용자로부터 행과 열의 개수를 입력받아 해당 행-열을 갖는 2차원 배열을 만들고 0 또는 1 중 임의의 값으로 채운 후 출력해 보라.
  - int** 변수를 사용하여 행의 개수만큼 int* 변수를 만들고
  - 각 int* 변수를 사용하여 열의 개수만큼 만든다.
  - 그 다음부터는 int** 변수를 2차원 배열과 동일하게 사용하면 된다.



프로그래밍 연습

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

void main(void)
{
    srand(time(NULL));
    int **Ary;
    int row, col;

    cout << "행과 열의 개수 입력 : ";
    cin >> row >> col;

    Ary = new int *[row];
    for (int i = 0; i < row; i++)
        Ary[i] = new int[col];

    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            Ary[i][j] = rand() % 2;
            cout << Ary[i][j] << " ";
        }
        cout << endl;
    }

    for (int i = 0; i < row; i++)
        delete [] Ary[i];
    delete [] Ary;
}
```

int * 배열 생성

각각의 int * 를 이용해 int 배열 생성

int ** 를 2차원 배열처럼 사용

메모리 해제