6강. 함수와 배열, 포인터, 참조

C++ 프로그래밍

jhhwang@kumoh.ac.kr

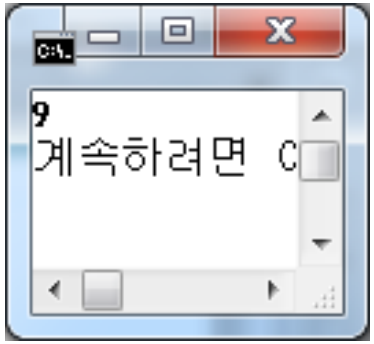
컴퓨터공학과 황준하

목차

함수와 포인터

- C++ 매개변수 전달 방법
  - 값에 의한 전달 : 변수값, 주소값
  - 참조에 의한 전달 : 참조변수
- 변수 주소값의 매개변수 전달

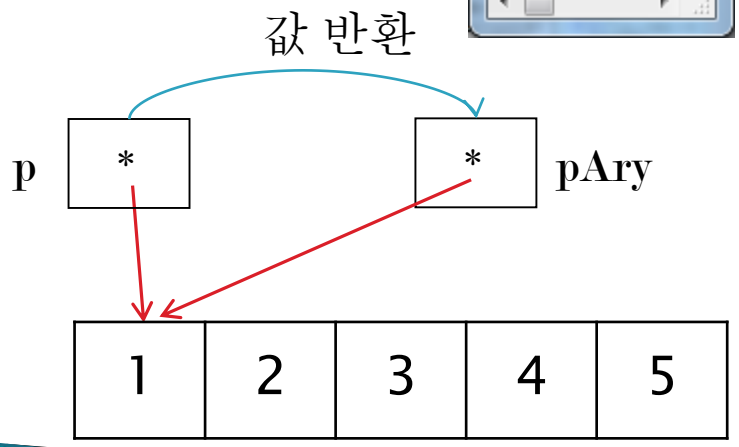
```
void Power(int *pNum)
{
	*pNum = *pNum * *pNum;
}

void main(void)
{
	int Num = 3;
	Power(&Num);
	cout << Num << endl;
}
```



pNum을 통해 Num 접근 가능

함수와 포인터

- 주소값의 반환
  - 주소값 반환도 가능
  - 단, 해당 타입의 포인터로 받음




```
int *GetArray(int Count)
{
	int *p = new int[Count];
	return p;
}

void main(void)
{
	int *pAry;
	pAry = GetArray(5);

	for (int i = 0; i < 5; i++) {
		pAry[i] = i + 1;
		cout << pAry[i] << endl;
	}

	delete [] pAry;
}
```

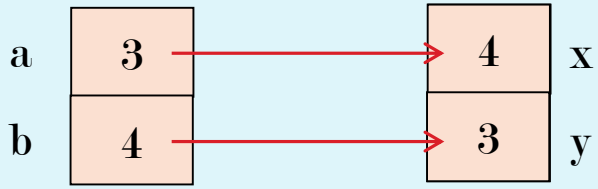
사용하고 난 후 메모리 해제

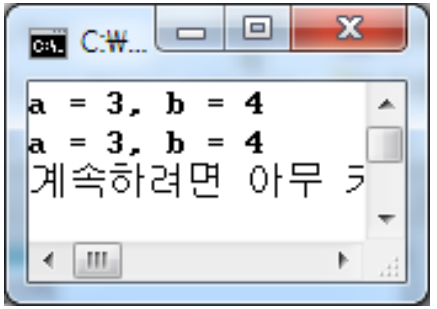
연습 문제

- 두 변수의 값을 서로 교환하는 함수를 다음과 같이 만들었더니 제대로 동작하지 않았다. 두 변수의 값이 교환되도록 프로그램을 수정하라.

```
void swap(int x, int y)
{
      int temp = x;
      x = y;
      y = temp;
}

void main(void)
{
      int a = 3, b = 4;
      cout << "a = " << a << ", b = " << b << endl;
      swap(a, b);
      cout << "a = " << a << ", b = " << b << endl;
}
```

변수값 전달의 문제점



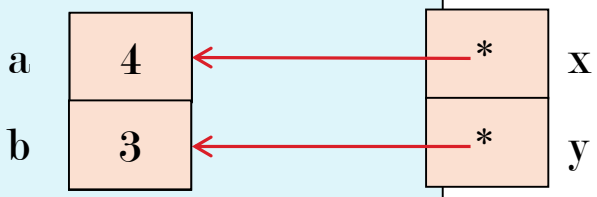


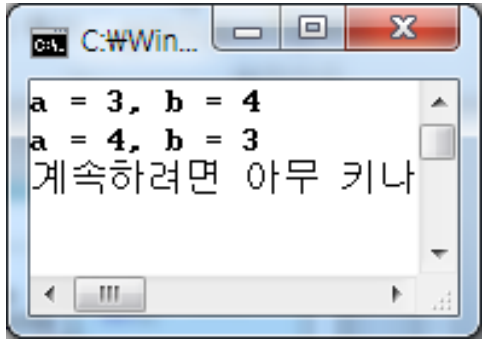
연습 문제

- 프로그램 확인

```
void swap(int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

void main(void)
{
    int a = 3, b = 4;
    cout << "a = " << a << ", b = " << b << endl;
    swap(&a, &b);
    cout << "a = " << a << ", b = " << b << endl;
}
```

해결 방법 : 주소값 전달

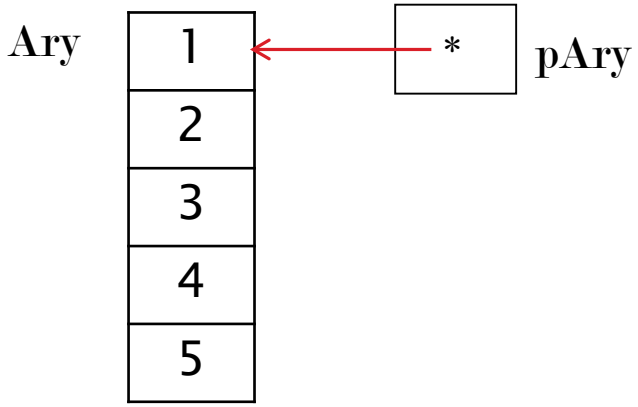




함수와 배열

참고 사항
int Ary[5];
Ary의 타입 : int *
&Ary의 타입 : int (*)[5]

- 배열의 매개변수 전달 방법
  - 첫 번째 원소의 주소값 전달
  - 원소의 개수를 함께 전달



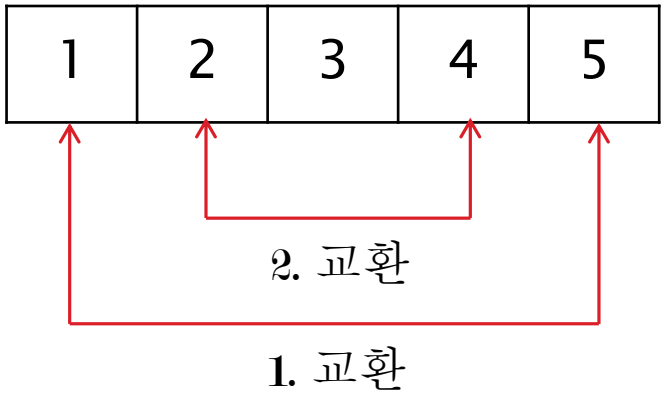

```
int Sum(int *pAry, int Count)
{
	int S = 0;
	for (int i = 0; i < Count; i++)
		S += pAry[i];
	return S;
}

void main(void)
{
	int Ary[5] = { 1, 2, 3, 4, 5 };
	cout << Sum(Ary, 5) << endl;
}
```

포인터를 배열처럼 사용

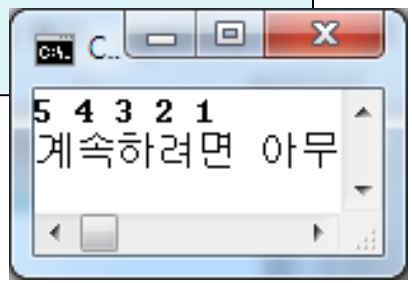
연습 문제

- int형 1차원 배열을 매개변수로 전달받아 원소들의 값을 역순으로 재정렬하는 함수 Reverse를 작성하라.

중간까지 반대편 원소와 교환




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

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

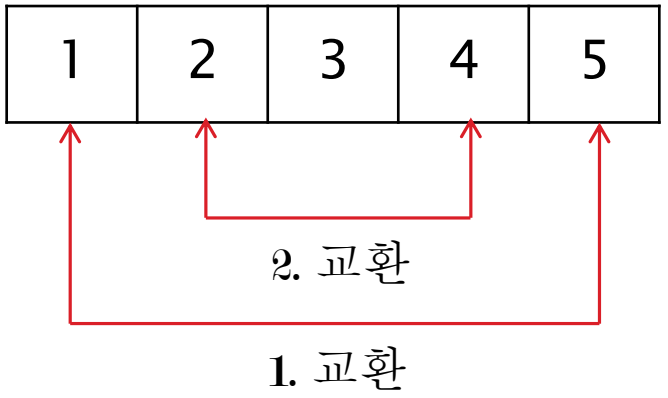


연습 문제

- 프로그램 확인

```
void Reverse(int *pAry, int Count)
{
	for (int i = 0; i < Count / 2; i++) {
		int temp = pAry[i];
		pAry[i] = pAry[Count - i - 1];
		pAry[Count - i - 1] = temp;
	}
}
```

중간까지 반대편 원소와 교환

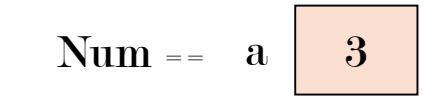


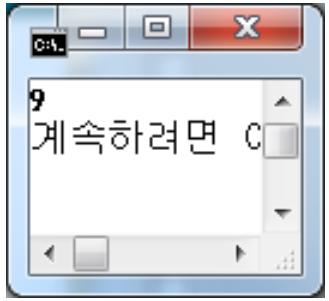
함수와 참조

- 참조에 의한 매개변수 전달
  - 변수 그 자체를 매개변수로 전달받을 수 있음

```
void square(int &Num)
{
	Num = Num * Num;
}

void main(void)
{
	int a = 3;
	square(a);
	cout << a << endl;
}
```

a와 Num은 동일한 변수





함수와 참조

- 참조의 반환
  - 변수 그 자체를 반환할 수 있음
  - 사라질 변수(지역변수와 같은) 그 자체를 반환하지 않도록 주의

```
int &GetNumber(int &Num)
{
	Num = Num + Num;
	return Num;
}

void main(void)
{
	int a = 3;
	GetNumber(a) = 100;
	cout << a << endl;
}
```

매개변수 전달

Num == a 6

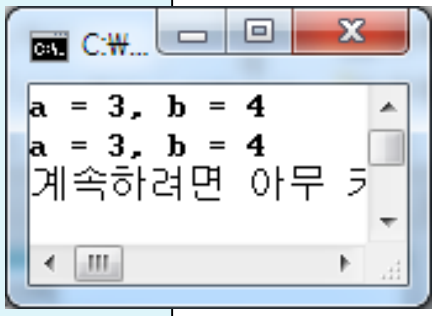
참조 반환

GetNumber(a) == Num == a 100

연습 문제

- 두 변수의 값을 서로 교환하는 함수인 swap 함수를 포인터가 아닌 참조를 사용하여 구현해 보라.

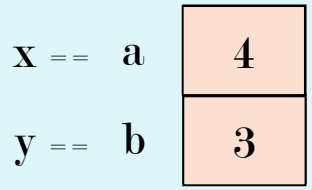
```
void swap(int x, int y)
{
	int temp = x;
	x = y;
	y = temp;
}

void main(void)
{
	int a = 3, b = 4;
	cout << "a = " << a << ", b = " << b << endl;
	swap(a, b);
	cout << "a = " << a << ", b = " << b << endl;
}
```

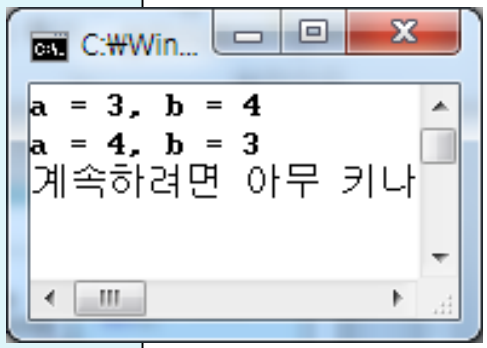


연습 문제

- 프로그램 확인

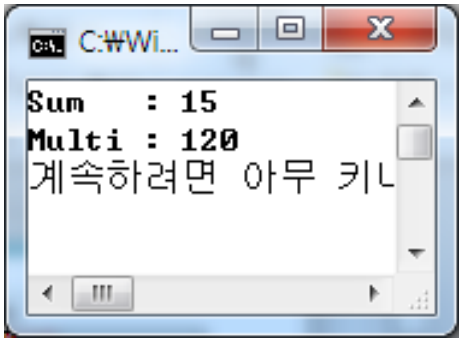
```
void swap(int &x, int &y)
{
    int temp = x;
    x = y;
    y = temp;
}

void main(void)
{
    int a = 3, b = 4;
    cout << "a = " << a << ", b = " << b << endl;
    swap(a, b);
    cout << "a = " << a << ", b = " << b << endl;
}
```





프로그래밍 연습

- int형 배열과 원소의 개수를 매개변수로 전달받아 원소들 전체를 더한 결과와 곱한 결과를 한꺼번에 반환하고자 한다. 그러나 함수의 반환값으로는 단 한 개만 반환이 가능하다. 매개변수를 사용하여 이를 해결해 보라.
  1. 포인터를 사용해 보라
  2. 참조를 사용해 보라.




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

    Sum = SumMulti(Ary, 5, Multi);   // ?

    cout << "Sum   : " << Sum << endl;
    cout << "Multi : " << Multi << endl;
}
```

프로그래밍 연습

- 포인터를 사용한 경우

```
int SumMulti(int *Ary, int Count, int *Mul)
{
	int Sum = 0;
	*Mul = 1;

	for (int i = 0; i < Count; i++) {
		Sum += Ary[i];
		*Mul *= Ary[i];
	}

	return Sum;
}
```

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

	Sum = SumMulti(Ary, 5, &Multi);

	cout << "Sum   : " << Sum << endl;
	cout << "Multi : " << Multi << endl;
}
```

프로그래밍 연습

- 참조를 사용한 경우

```
int SumMulti(int *Ary, int Count, int &Mul)
{
	int Sum = 0;
	Mul = 1;

	for (int i = 0; i < Count; i++) {
		Sum += Ary[i];
		Mul *= Ary[i];
	}

	return Sum;
}
```

Mul과 Multi는 동일

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

	Sum = SumMulti(Ary, 5, Multi);

	cout << "Sum   : " << Sum << endl;
	cout << "Multi : " << Multi << endl;
}
```