Practice 10

경희대학교
KYUNG HEE UNIVERSITY

Practice 10. Array and Pointer 2

Problem 1. Make a "max" programs.

Objective: Implement a program that prompt a user 3 integers and calculate the sum of the biggest and smallest integers

Conditions:

1. Make the following three functions.
2. The min and max returns the pointer of smallest and biggest integer, respectively.
3. The sum function calculate the results using the two pointers.
4. Display the result as follows.

min function

- Prototype : int* min(int *pfirst, int *pSecond, int *pThird);
- return the pointer of the smallest integer among the values that the three pointer parameters point at

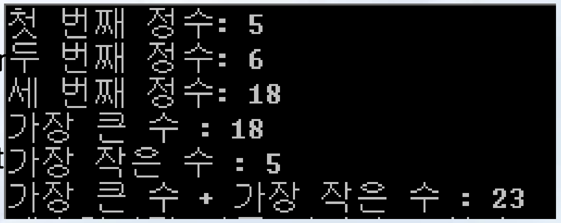
max function

- Prototype : int* max(int* pFirst, int* pSecond, int* pThird);
- return the pointer of the biggest integer among the values that the three pointer parameters point at

sum 함수

- Prototype : void sum(int* pInt1, int* pInt2, int* pResult);
- Calculate the sum of the values of the two input pointers and store it to the memory that pResult points at

Result example



First integer	첫 번째 정수: 5
Second integer	두 번째 정수: 6
Third integer	세 번째 정수: 18
The biggest	가장 큰 수 : 18
The smallest	가장 작은 수 : 5
The sum	가장 큰 수 + 가장 작은 수 : 23

Practice 10. Array and Pointer 2

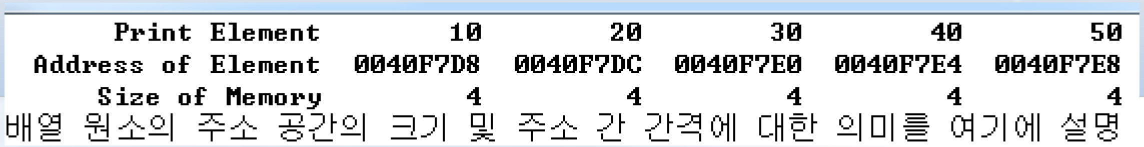
• Problem 2

Objective: Understanding the address of an array

- Initialize an array, print the address of it the using the following code, and understand how the address of an array is mapped

```
Int main(){
    int Ary[5] = {10, 20, 30, 40, 50};
    cout << setw(20) << "Print Element";
    for( int i=0; i<5; i++ )
        cout << setw(10) << (__________) ;      // Print the elements of the array using index operator []
    cout << endl;

    cout <<  setw(20) << "Address of Element";
    for( int i=0; i<5; i++ )
        cout << setw(10) << (__________);      // Print the address of the elements using [] and &

    cout <<  setw(30) << "Size of Memory";
    for( int i=0; i<5; i++ )
        cout << setw(10) <<  (__________);      // Print the size of each element using sizeof() function
    cout << endl;

    char *answer =  "______________________(Write your understanding for the memory address and the 
size)";
    cout << answer << endl;
    return 0;
}
```

output



```
     Print Element        10        20        30        40        50
Address of Element  0040F7D8  0040F7DC  0040F7E0  0040F7E4  0040F7E8
    Size of Memory         4         4         4         4         4
배열 원소의 주소 공간의 크기 및 주소 간 간격에 대한 의미를 여기에 설명
```

Practice 10. Array and Pointer 2

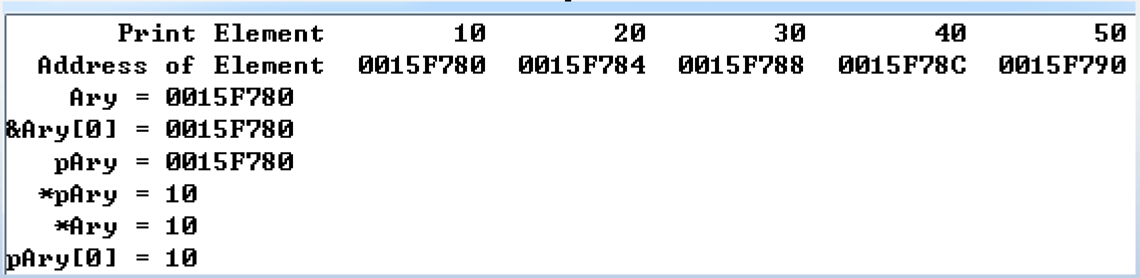
- **Problem 3**

Objective: Array name and pointer

- Add the following code segment to the problem 2, in order to print the memory address stored in the array name and the address of the first element.
- Also initialize an integer pointer using the address of an array, and print the value of that pointer

```
 // Problem 2’ s code
 cout << setw(10) << "Ary = " << (________) << endl;      // Print the name of the array
 cout << setw(10) << "&Ary[0] = " << (________) << endl; // Print the address of the first element

 int* pAry = (________);                                  // Initialize an integer pointer with the array name
 cout << setw(10) << "pAry = " << (________) << endl;     // Print the pointer
 cout << setw(10) << "*pAry = " << (________) << endl;    // Access the first element using * and the pointer
 cout << setw(10) << "*Ary = " << (________) << endl;     // Access the first element using * and the array
name
 cout << setw(10) << "pAry[0] = " << (________) << endl; // Access the first element using [] and the pointer
```

output



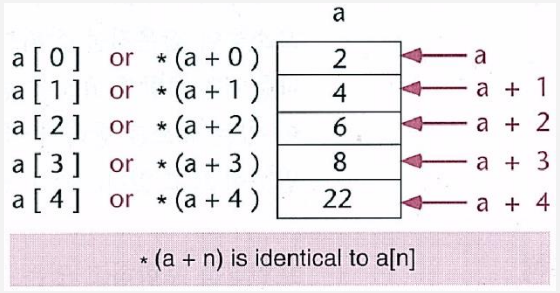
```
    Print Element        10        20        30        40        50
  Address of Element 0015F780  0015F784  0015F788  0015F78C  0015F790
     Ary = 0015F780
&Ary[0] = 0015F780
    pAry = 0015F780
   *pAry = 10
    *Ary = 10
pAry[0] = 10
```

Practice 10. Array and Pointer 2

- **Problem 4**

Pointer arithmetic and array – Offset calculation

- Using the following program code, access the elements of the array only using the pointer arithmetic and offset calculation

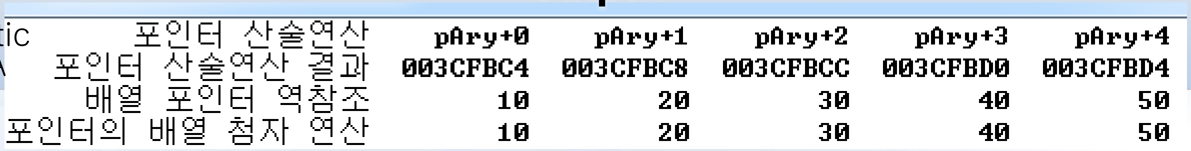


```
int Ary[5] = {10, 20, 30, 40, 50};
int* pAry = Ary;

// Displaying using pointer arithmetic
cout << setw(23) << "Pointer arithmetic";
for( int k=0; k<5; k++ )
     cout << setw(9) << "pAry+" << k;
cout << endl;

cout << setw(23) << "The result of PA";
(_______) //Print the address of each element using PA
```

```
cout << setw(23) << "Value using *";
// (_______) Print the value of each element using *
cout << setw(23) << "Value using []";
// (_______) Print the value using []
```

output

Pointer arithmetic
The result of PA
Value using *
Value using []



```
         포인터 산술연산   pAry+0   pAry+1   pAry+2   pAry+3   pAry+4
     포인터 산술연산 결과 003CFBC4 003CFBC8 003CFBCC 003CFBD0 003CFBD4
        배열 포인터 역참조       10       20       30       40       50
포인터의 배열 첨자 연산       10       20       30       40       50
```

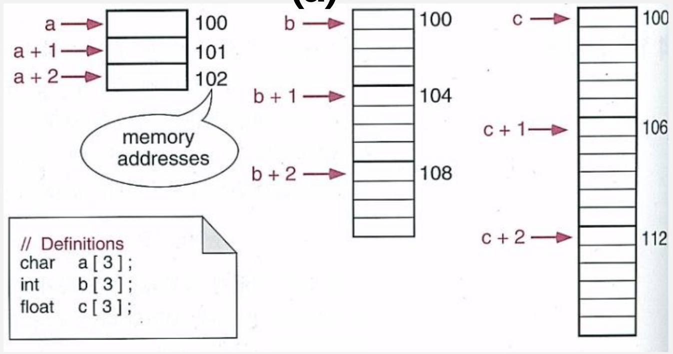
Practice 10. Array and Pointer 2

• Break time

Different offset size for different type

- (a) illustrates different offset size for different type (assuming char 1byte, int 4byte, float 6byte). Using a debugging tool, the memory address of the variables in the sample code (b) can be printed (see (c)). Examine the result.

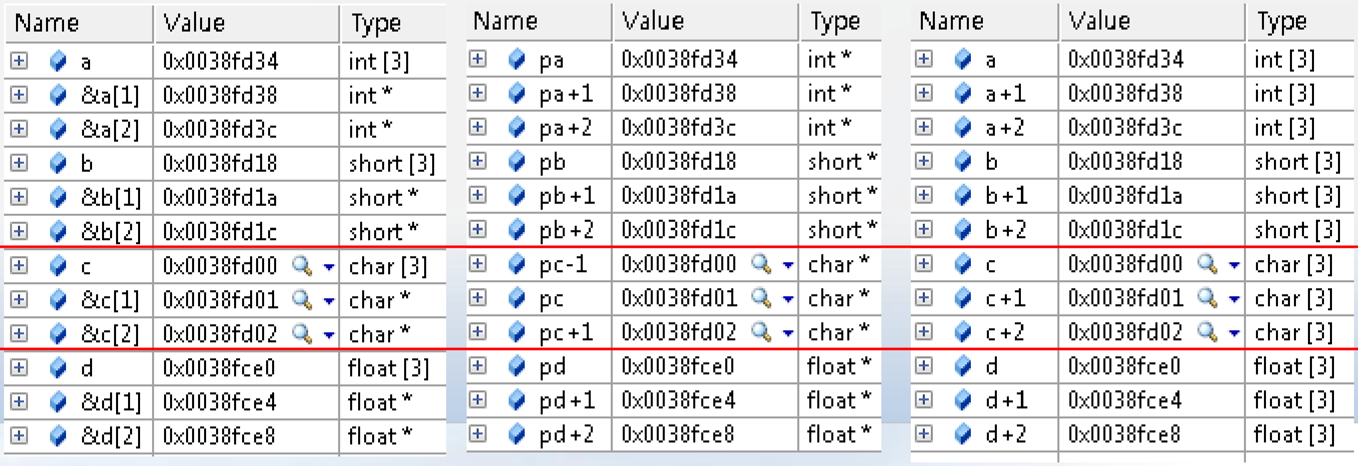
(a)



(b)

```
// 샘플 코드
int a[3];          int* pa = a;
short int b[3];    short int* pb = b;
char c[3];         char* pc = &c[1];
float d[3];        float* pd = d;
```

(c)



pa = a

pb = b

pc = &c[1]

pd = d

Name	Value	Type
a	0x0038fd34	int [3]
&a[1]	0x0038fd38	int *
&a[2]	0x0038fd3c	int *
b	0x0038fd18	short [3]
&b[1]	0x0038fd1a	short *
&b[2]	0x0038fd1c	short *
c	0x0038fd00	char [3]
&c[1]	0x0038fd01	char *
&c[2]	0x0038fd02	char *
d	0x0038fce0	float [3]
&d[1]	0x0038fce4	float *
&d[2]	0x0038fce8	float *

Name	Value	Type
pa	0x0038fd34	int *
pa+1	0x0038fd38	int *
pa+2	0x0038fd3c	int *
pb	0x0038fd18	short *
pb+1	0x0038fd1a	short *
pb+2	0x0038fd1c	short *
pc-1	0x0038fd00	char *
pc	0x0038fd01	char *
pc+1	0x0038fd02	char *
pd	0x0038fce0	float *
pd+1	0x0038fce4	float *
pd+2	0x0038fce8	float *

Name	Value	Type
a	0x0038fd34	int [3]
a+1	0x0038fd38	int [3]
a+2	0x0038fd3c	int [3]
b	0x0038fd18	short [3]
b+1	0x0038fd1a	short [3]
b+2	0x0038fd1c	short [3]
c	0x0038fd00	char [3]
c+1	0x0038fd01	char [3]
c+2	0x0038fd02	char [3]
d	0x0038fce0	float [3]
d+1	0x0038fce4	float [3]
d+2	0x0038fce8	float [3]

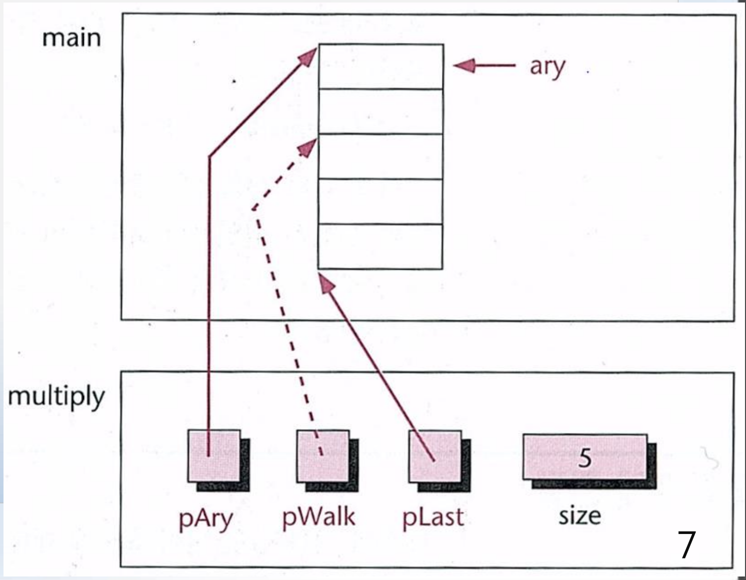
Float was 4byte

Practice 10. Array and Pointer 2

- **Problem 5**

Objective: Array as a function parameter

- Make multiplyPtr( ) function that gets an array as a parameter and doubles each elements of the array.
  - The elements must be accessed only using pAry, pWalk, pLast as shown below
  - pAry : pointing the first element, pWalk : pointer for traverse, pLast : pointing the last element
  - ex) pLast = pAry + size −1;
- In the main( ), call multiplyPrt( ) and print the result array
- Use checkSize( ) to test how the compiler deal with the size of an array in the main and in the function.
- Refer to the code in the next slide

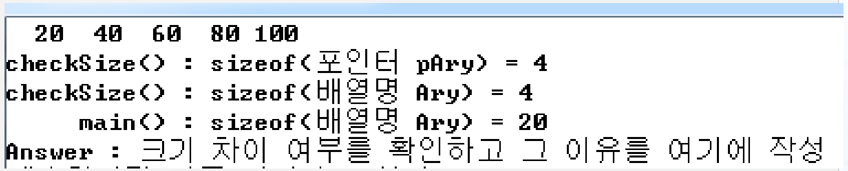


Practice 10. Array and Pointer 2

- **Problem 5**

```
void multiplyPtr(int* pAry, int size);

// Gets an array and print the size of the array using sizeof()
void checkSize(int* pAry, int Ary[]);

int main(){
     int Ary[ ] = { 10, 20, 30, 40, 50};
     int* pAry = Ary;
     int arySize = sizeof(Ary)/sizeof(Ary[0]);

     multiplyPtr(Ary, arySize);
     (________)      //Print the result

     checkSize(pAry, Ary, cSize);
     cout << setw(35) << "main() : sizeof(배열명 Ary) = " << sizeof(Ary) << endl;
     char* answer = "________________________ (Explain why there is a difference in size)";
     cout << answer << endl;
     return 0;
}
void multiplyPtr(int* pAry, int size){ ________ }
void checkSize(int* pAry, int Ary[], int size){
     cout << "checkSize() : sizeof(포인터 pAry) = " << sizeof(pAry) << endl;
     cout << "checkSize() : sizeof(배열명 Ary) = " << sizeof(Ary) << endl;
}
```

output



```
  20   40   60   80 100
checkSize() : sizeof(포인터 pAry) = 4
checkSize() : sizeof(배열명 Ary) = 4
     main() : sizeof(배열명 Ary) = 20
Answer : 크기 차이 여부를 확인하고 그 이유를 여기에 작성
```

Practice 10. Array and Pointer 2

- **Problem 6. Binary Search Using Pointers**

In Program 9-11 (page 449), a binary search is implemented using pointers.

Understand this function and implement the main function that calls this function.

In the Main, make an array of size 100, and fill it with random number ranged from 1 to 100.

Get a user's integer input and do binary search using the user's input.

Note that the array should be sorted in order to apply binary search.