

2013 Fall Semester SMP

Day 3 – Array, Function, Conditional Stmt., For Stmt.

Data type with I/O

Function

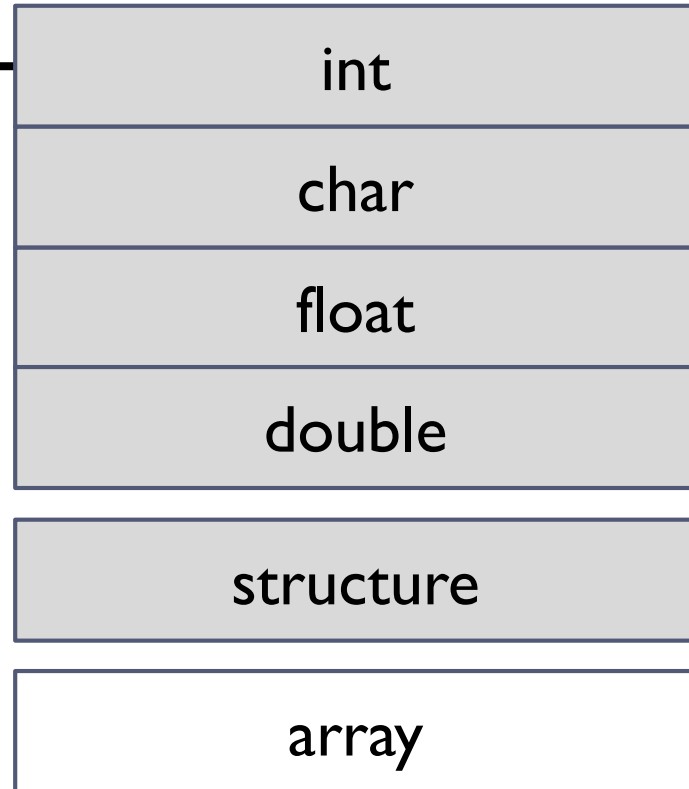
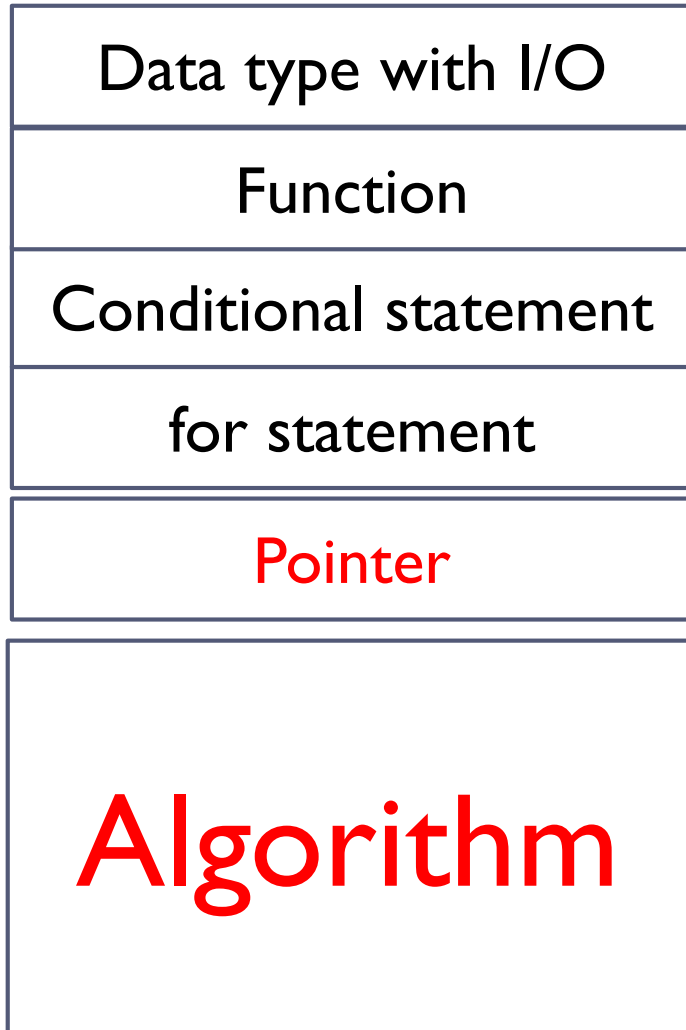
Conditional statement

for statement

Pointer

Algorithm





Array

- ▶ 여러 개의 변수를 동시에 선언할 수 있는 방법
 - ▶ ex) 10개 변수 선언하기
`int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10;`
 - ▶ 벨짓...
- ▶ `int Array[10];`
 - ▶ 0번째부터 시작함

	0	1	2	3	4	5	6	7	8	9
Array										



Array

- ▶ 여러 개의 변수를 동시에 선언할 수 있는 방법
 - ▶ ex) 10개 변수 선언하기
`int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10;`
 - ▶ 빨갓...
- ▶ `int Array[10];`
 - ▶ 0번째부터 시작함
 - ▶ `Array[3] = 4;`

	0	1	2	3	4	5	6	7	8	9
Array										



Array

- ▶ 여러 개의 변수를 동시에 선언할 수 있는 방법
 - ▶ ex) 10개 변수 선언하기
`int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10;`
 - ▶ 벨짓...
- ▶ `int Array[10];`
 - ▶ 0번째부터 시작함
 - ▶ `Array[3] = 4;`

	0	1	2	3	4	5	6	7	8	9
Array				4						



Array

- ▶ N차원 Array를 선언할 수 있음
 - ▶ 1차원 Array : `int Array1D[5];`
 - ▶ 2차원 Array : `int Array2D[5][5];`
 - ▶ n차원 Array : `int ArraynD[5][5]...[5];` (n번)

	0	1	2	3	4
0					
1					
2					
3					
4					



Array

- ▶ N차원 Array를 선언할 수 있음
 - ▶ 1차원 Array : `int Array1D[5];`
 - ▶ 2차원 Array : `int Array2D[5][5];`
 - ▶ n차원 Array : `int ArraynD[5][5]...[5];` (n번)

	0	1	2	3	4
0					
1					
2					
3					
4					

`Array2D[1][2] = 3;`



Array

- ▶ N차원 Array를 선언할 수 있음
 - ▶ 1차원 Array : `int Array1D[5];`
 - ▶ 2차원 Array : `int Array2D[5][5];`
 - ▶ n차원 Array : `int ArraynD[5][5]...[5];` (n번)

	0	1	2	3	4
0					
1			3		
2					
3					
4					

`Array2D[1][2] = 3;`



Example

- ▶ 3 쌍의 숫자를 입력 받아 각각의 합을 출력

Input

1 2

2 3

3 6

Output

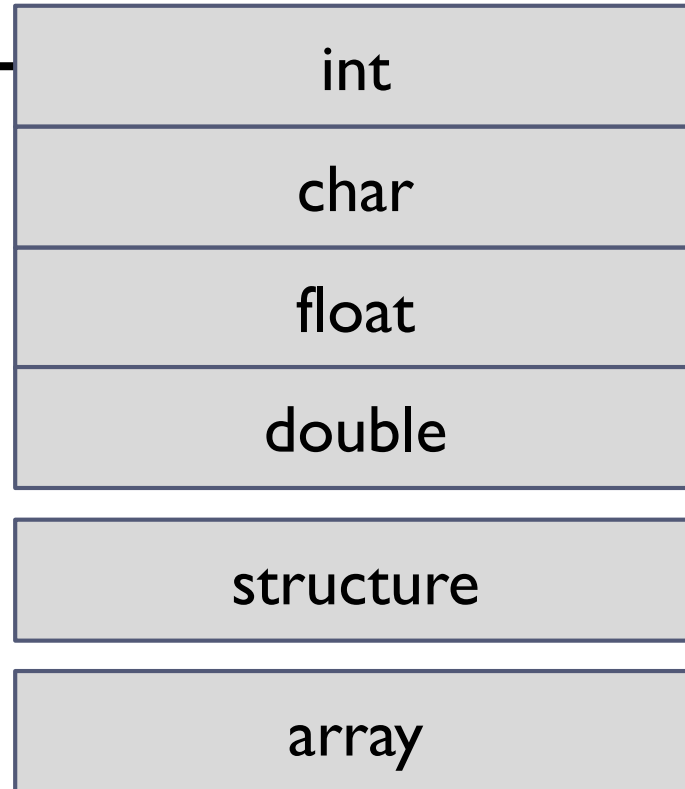
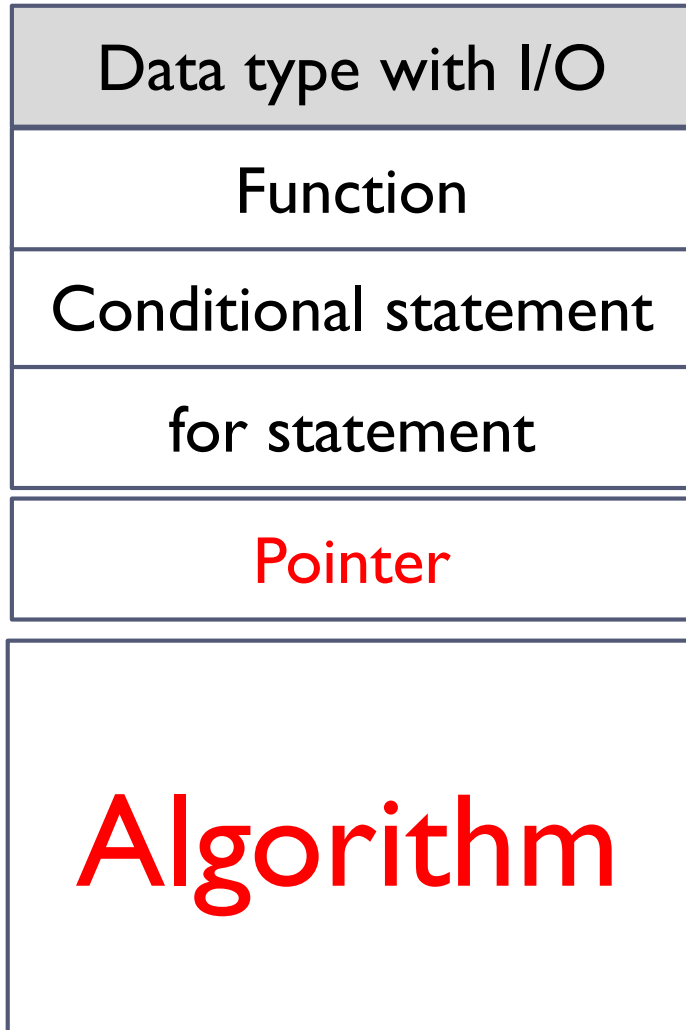
3 6 9



Array

- ▶ 어떻게 구현되어 있나?
 - ▶ **Pointer**
 - ▶ 약간 골치가 아파집니다...





Data type with I/O

Function

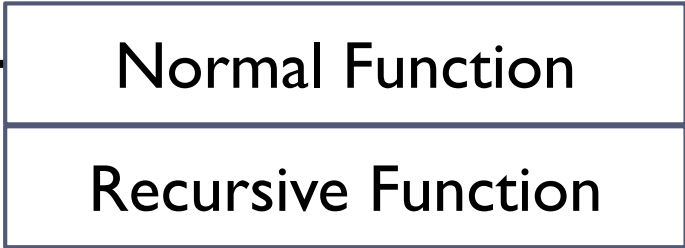
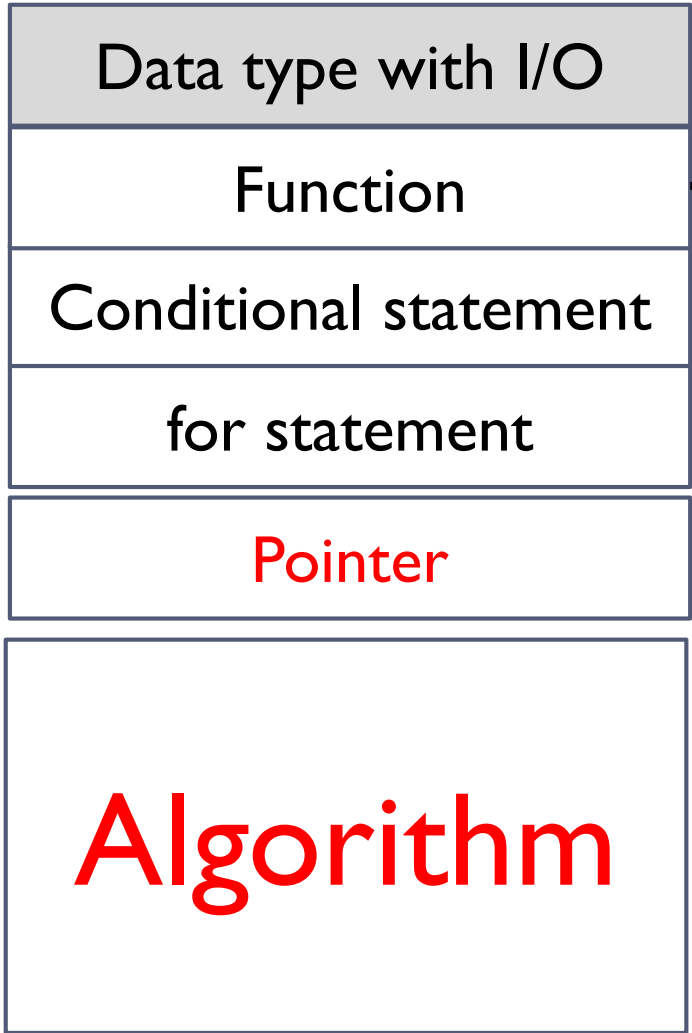
Conditional statement

for statement

Pointer

Algorithm





Function

- ▶ $y = f(x)$ 같은 함수 맞춤
 - ▶ 어떤 값을 넣으면 어떤 값이 튀어나오는 것
 - ▶ 값은 여러 개 넣을 수 있지만 단 하나의 값만 튀어나옴
- ▶ Function을 잘 써서 코딩을 하면 코드가 이뻐짐
 - ▶ 간결하고!
 - ▶ 이해하기 쉽고!



Function

```
#include <stdio.h>
#include <algorithm>
using namespace std;
void function(int x, int y){
    printf("%d %d\n",x, y);
}
int main(){
    function(1, 2);
    return 0;
}
```



Function

```
#include <stdio.h>
```

```
#include
```

```
using
```

```
void f
```

```
    pr
```

```
}
```

```
int ma
```

```
    fu
```

```
    re
```

```
}
```



C:\WgVimPortableWAppWvimWvim72Wvimrun.exe

C:\Windows\system32\cmd.exe /c "a.exe"

1 2

Hit any key to close this window...

-



Function

```
void function(int x, int y)
```



Function

void function(int x, int y)

반환하고자 하는 Type



Function

```
void function(int x, int y)
```

함수 이름



Function

`void function(int x, int y)`

함수에게 넘겨주고자 하는 값



Function

```
#include <stdio.h>
#include <algorithm>
using namespace std;
void function(int x, int y){
    printf("%d %d\n",x, y);
}
int main(){
    function(1, 2);
    return 0;
}
```



Function

- ▶ `add(x, y)` : `x`와 `y`를 더해서 출력하는 함수
 - ▶ How to design it?

```
void add(int x, int y) {  
    printf("%d\n",x+y);  
}
```



Function

- ▶ `add(x, y)` : `x`와 `y`를 더해서 이를 반환하는 함수
 - ▶ `int x = add(1, 3);` // 이런 것이 가능하게끔
 - ▶ How to design it?



Function

- ▶ `add(x, y)` : `x`와 `y`를 더해서 이를 반환하는 함수
 - ▶ `int x = add(1, 3);` // 이런 것이 가능하게끔
 - ▶ How to design it?

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



Function

```
#include <stdio.h>
#include <algorithm>
using namespace std;
int add(int x, int y){
    return x+y;
}
int main(){
    int a, b;
    scanf("%d %d",&a, &b);
    printf("%d\n",add(a, b));
    return 0;
}
```

를 반환하는 함수
능하게끔

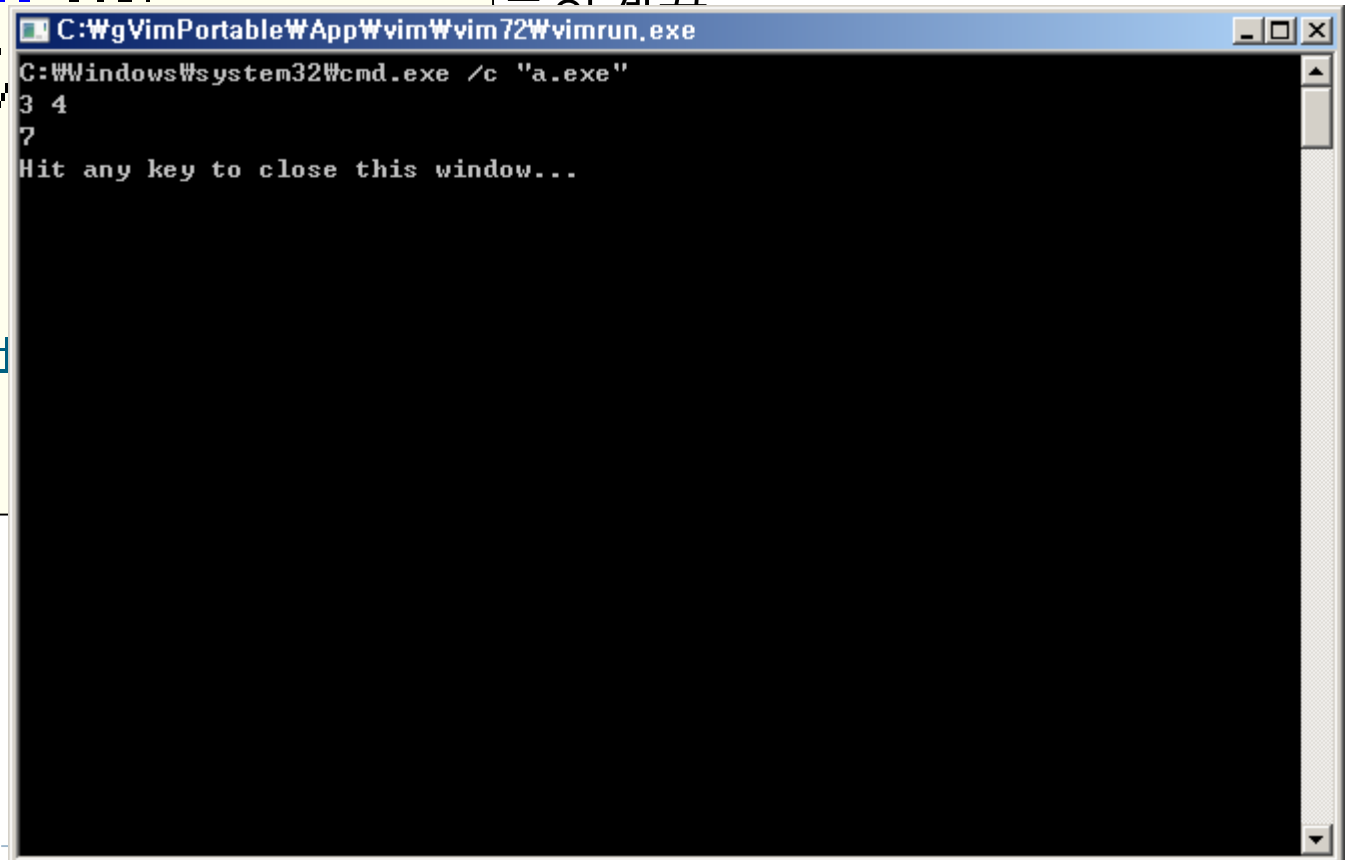


Function

```
#include <stdio.h>
#include <algorithm>
using namespace std;
int add(int x,
        return x+y
}
int main(){
    int a, b;
    scanf("%d
printf("%d
return 0;
}
```

를 반환하는 함수

는 실행



```
C:\WgVimPortableWAppWvimWvim72Wvimrun.exe
C:\Windows\system32\cmd.exe /c "a.exe"
3 4
7
Hit any key to close this window...
```

Function

▶ Function의 치명적인 단점

- ▶ main에서 넘겨 주는 것은 변수 자체가 아니라 값만 넘겨줌
- ▶ Function 내에서 무슨 짓을 하던 간에 main에서는 영향 X

Ex. Swap(a, b) : 변수 a와 변수 b의 값을 서로 바꾸어 주는 함수



Function

▶ Function의 치명적인 단점

- ▶ main에서 넘겨 주는 것은 변수 자체가 아니라 값만 넘겨줌
- ▶ Function 내에서 무슨 짓을 하던 간에 main에서는 영향 X

Ex. Swap(a, b) : 변수 a와 변수 b의 값을 서로 바꾸어 주는 함수

```
void Swap(int a, int b){  
    int temp = a;  
    a = b;  
    b = temp;  
}
```



Function

▶ Function의 치명적인 단점

- ▶ main에서 넘겨 주는 것은 변수 자체가 아니라 값만 넘겨줌
- ▶ Function 내에서 무슨 짓을 하던 간에 main에서는 영향 X

Ex. Swap(a, b) : 변수 a와 변수 b의 값을 서로 바꾸어 주는 함수

```
void Swap(int a, int b){  
    int temp = a;  
    a = b;  
    b = temp;  
}
```

```
int main(){  
    int x = 3, y = 2;  
    Swap(x, y);  
    printf("x = %d, y = %d", x, y);  
    return 0;  
}
```



Function

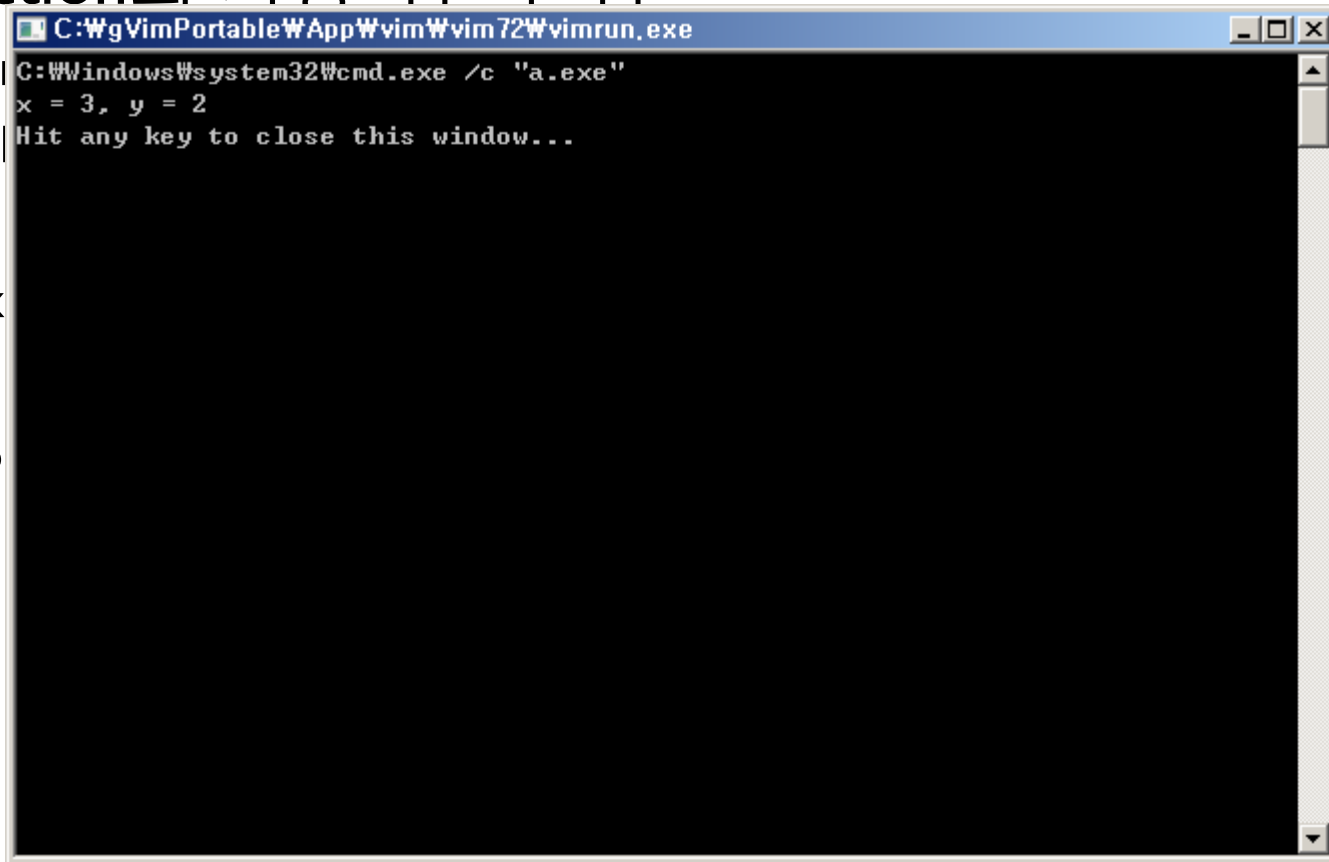
▶ Function의 치명적인 단점

▶ C:\Windows\system32\cmd.exe /c "a.exe"
x = 3, y = 2
▶ Hit any key to close this window...

Ex

VO

}



The screenshot shows a Windows command prompt window with the title bar "C:\WgVimPortable\App\vim\vim72\vimrun.exe". The command prompt displays the following text: "C:\Windows\system32\cmd.exe /c \"a.exe\"", "x = 3, y = 2", and "Hit any key to close this window...". The window has a black background and a white border.



Function

- ▶ **Function의 치명적인 단점**
 - ▶ main에서 넘겨 주는 것은 변수 자체가 아니라 값만 넘겨줌
 - ▶ Function 내에서 무슨 짓을 하던 간에 main에서는 영향 X
- ▶ **How to solve it?**
 - ▶ Using **pointer!**

Swap(a, b) : a가 가리키는 애와 b가 가리키는 애의 값을 바꿈



Function

- ▶ Function의 치명적인 단점
 - ▶ main에서 넘겨 주는 것은 변수 자체가 아니라 값만 넘겨줌
 - ▶ Function 내에서 무슨 짓을 하던 간에 main에서는 영향 X
- ▶ How to solve it?
 - ▶ Using **pointer!**

Swap(a, b) : a가 가리키는 애와 b가 가리키는 애의 값을 바꿈

```
void Swap(int* a, int* b){  
    int temp;  
    temp = *a;  
    *a = *b;  
    *b = temp;  
}
```



Function

- ▶ Function의 치명적인 단점
 - ▶ main에서 넘겨 주는 것은 변수 자체가 아니라 값만 넘겨줌
 - ▶ Function 내에서 무슨 짓을 하던 간에 main에서는 영향 X
- ▶ How to solve it?
 - ▶ Using **pointer!**

Swap(a, b) : a가 가리키는 애와 b가 가리키는 애의 값을 바꿈

```
void Swap(int* a, int* b){  
    int temp;  
    temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

```
int main(){  
    int x = 3, y = 2;  
    ????????  
    printf("x = %d, y = %d\n", x, y);  
    return 0;  
}
```

Function

- ▶ Function의 치명적인 단점
 - ▶ main에서 넘겨 주는 것은 변수 자체가 아니라 값만 넘겨줌
 - ▶ Function 내에서 무슨 짓을 하던 간에 main에서는 영향 X
- ▶ How to solve it?
 - ▶ Using **pointer!**

Swap(a, b) : a가 가리키는 애와 b가 가리키는 애의 값을 바꿈

```
void Swap(int* a, int* b){  
    int temp;  
    temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

```
int main(){  
    int x = 3, y = 2;  
    Swap(&x, &y);  
    printf("x = %d, y = %d\n", x, y);  
    return 0;  
}
```

Function

▶ Function의 기본적인 단자

▶ m x = 2, y = 3

▶ Fu

▶ How

▶ U

Swa

voic

il

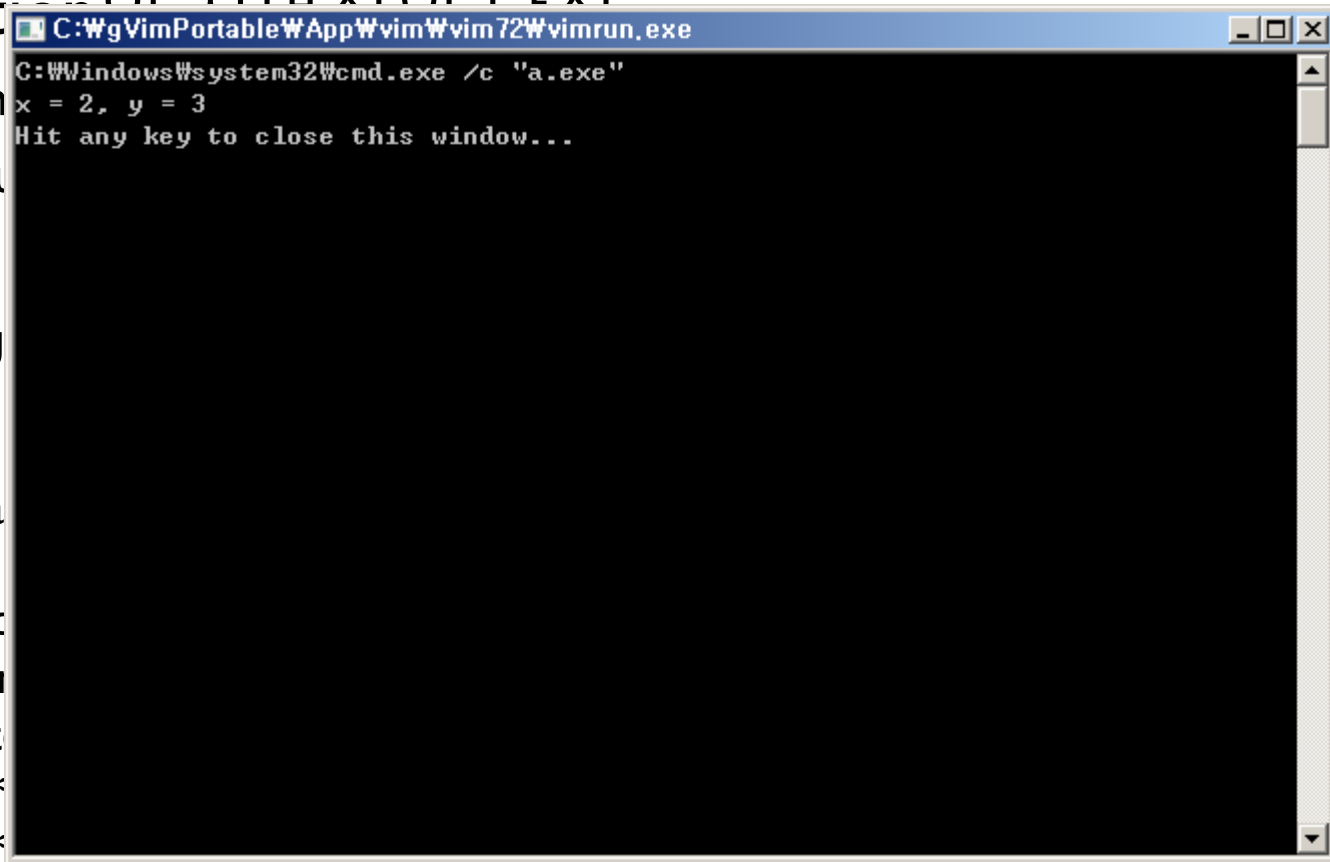
t

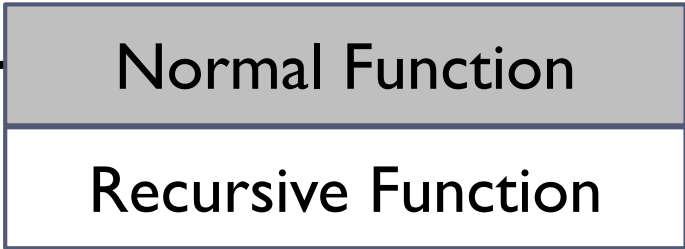
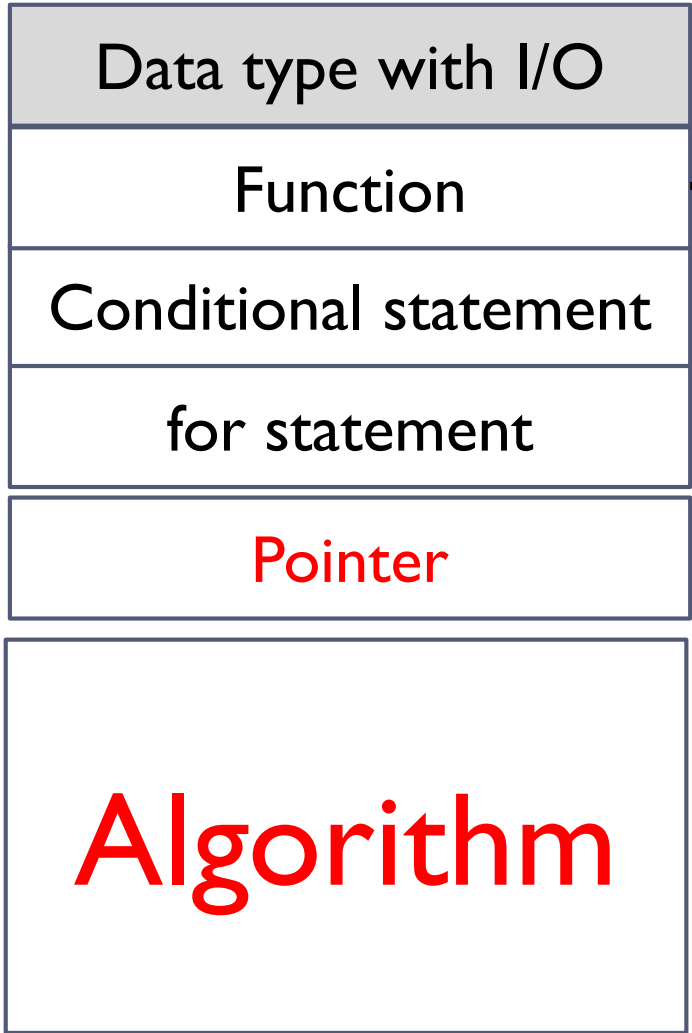
*

*

}

}





Data type with I/O

Function

Conditional statement

for statement

Pointer

Algorithm



Conditional Stmt.

▶ 조건의 참, 거짓에 따라 다르게 행동

어느 아내가 프로그래머 남편에게

「쇼핑하러 갈 때, 우유 하나 사와. 아, 계란 있으면 6개 사와」

남편은 잠시 후, 우유를 6개 사왔다.
아내는 물었다.

「왜 우유를 6개나 사왔어!」

남편 「계란이 있길래 6개 사왔지...」



Conditional Stmt.

▶ 조건의 참, 거짓에 따라 다르게 행동

어느 아내가 프로그래머 남편에게

「쇼핑하러 갈 때, 우유 하나 사와. 아, 계란 있으면 6개 사와」

남편은 잠시 후, 우유를 6개 사왔다.
아내는 물었다.

「왜 우유를 6개나 사왔어!」

남편 「계란이 있길래 6개 사왔지...」

```
int Number = 0;
if(ThereExist(Egg)) Number = 6;
else Number = 1;

Buy_Milk(Number);
```



Conditional Stmt.

- ▶ 대표적으로 두 개의 문법이 있다
 - ▶ if
 - ▶ switch-case
- ▶ if를 자주 씀
 - ▶ 편하고 직관적
 - ▶ switch-case 는 사실 잘 쓰지 않습니다.



Conditional Stmt.

```
if ( 조건 )  
{  
    ...  
}  
else if( 조건 )  
{  
    ...  
}  
else  
{  
    ...  
}
```



Conditional Stmt.

```
#include <stdio.h>
int main(){
    int a;
    printf("남자면 1, 여자면 2를 입력하세요 : ");
    scanf("%d",&a);
    if(a == 1)
    {
        printf("메효..#n");
    }
    else if(a == 2){
        printf("사랑해#n");
    }
    else{
        printf("???;.;#n");
    }
    return 0;
}
```



Conditional Stmt.

- ▶ 여러 가지 비교연산자가 있음
 - ▶ ==, !=, >=, <=, >, <
 - ▶ 직관적으로 알 수 있겠죠?
- ▶ 여러 조건을 논리연산자로 묶을 수 있음
 - ▶ &&, ||



Conditional Stmt.

- ▶ A가 2의 배수 혹은 3의 배수인지 판별 ?
 - ▶ `if(A % 2 == 0 || A % 3 == 0)`
- ▶ A가 1 ~ 100 사이의 짝수인지 판별 ?
 - ▶ `if((1 <= A && A <= 100) && A % 2 == 0)`
- ▶ 괄호를 잘 쓰길 바랍니다
 - ▶ `if(A % 2 == 0 || A % 3 == 0 && A % 7 == 0)`
 - ▶ 보는 사람도 헛갈리고 나도 헛갈리고 모두가 헛갈림



Conditional Stmt.

▶ switch – case

```
#include <stdio.h>
int main(){
    int x;
    scanf("%d",&x);
    switch(x){
        case 1 : printf("1"); break;
        case 2 : printf("2"); break;
        case 3 : printf("3"); break;
        default : printf(">= 4"); break;
    }
    printf("#n");
    return 0;
}
```



Conditional Stmt.

▶ switch – case

▶ break에 주의할 것

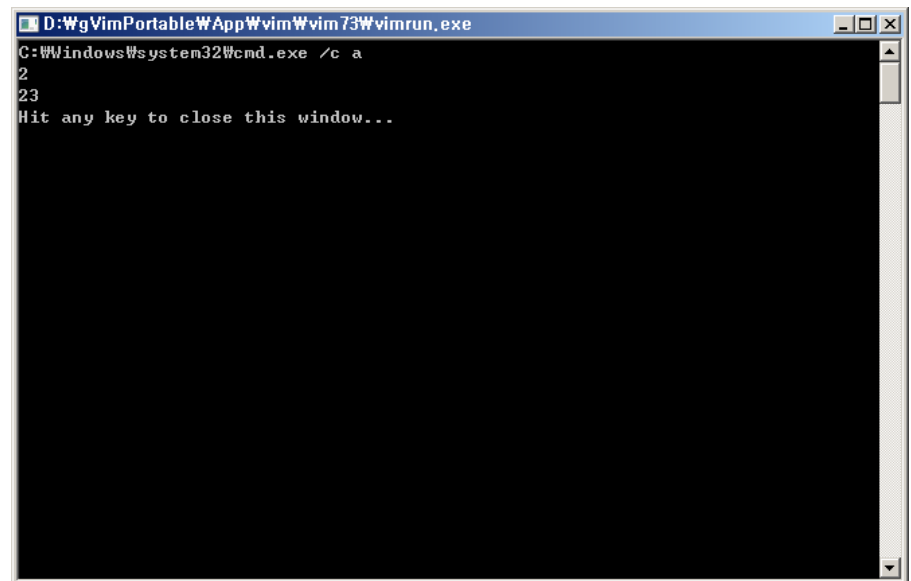
```
#include <stdio.h>
int main(){
    int x;
    scanf("%d",&x);
    switch(x){
        case 1 : printf("1");
        case 2 : printf("2");
        case 3 : printf("3"); break;
        default : printf(">= 4");
    }
    printf("#n");
    return 0;
}
```



Conditional Stmt.

- ▶ switch – case
 - ▶ break에 주의할 것

```
#include <stdio.h>
int main(){
    int x;
    scanf("%d",&x);
    switch(x){
        case 1 : printf("1");
        case 2 : printf("2");
        case 3 : printf("3"); break;
        default : printf(">= 4");
    }
    printf("#n");
    return 0;
}
```



```
D:\WgVimPortable\WApp\vim\vim73\vimrun.exe
C:\Windows\system32\cmd.exe /c a
2
23
Hit any key to close this window...
```



Data type with I/O

Function

Conditional statement

for statement

Pointer

Algorithm



For Stmt.

- ▶ 똑같은 일을 계속해서 반복하는 문법
 - ▶ while
 - ▶ do – while
 - ▶ for
- ▶ 매우 잘 다룰 줄 알아야 합니다
 - ▶ 처음 배울 때 진짜 어려움
 - ▶ 일단은 문법만 이해하기로...



For Stmt.

▶ While

- ▶ 조건이 참이면 반복, 거짓이면 종료

```
#include <stdio.h>
int main(){
    while(1){
        printf("Wow!\n");
    }
    return 0;
}
```

```
#include <stdio.h>
int main(){
    int t = 10;
    while(t > 0){
        printf("Wow!\n");
        t--;
    }
    return 0;
}
```



For Stmt.

▶ do – while

- ▶ 조건이 참이면 반복, 거짓이면 종료
- ▶ 일단 무조건 한번은 반복
- ▶ 거의 쓸 일 없으니 그냥 이런 것이 있구나... 합시다

```
#include <stdio.h>
int main(){
    int t = 10;
    do{
        printf("Wow!\n");
        t--;
    }while(t > 0);
    return 0;
}
```



For Stmt.

▶ for

- ▶ 제일 중요함
- ▶ 마찬가지로 조건이 참일 때 반복, 거짓이면 종료

for (초기값 ; 조건 ; 증가량)



For Stmt.

▶ for

- ▶ 제일 중요함
- ▶ 마찬가지로 조건이 참일 때 반복, 거짓이면 종료

```
#include <stdio.h>
int main(){
    int i;
    for(i=0;i<10;i++){
        printf("Wow! : %d\n", i);
    }
    return 0;
}
```

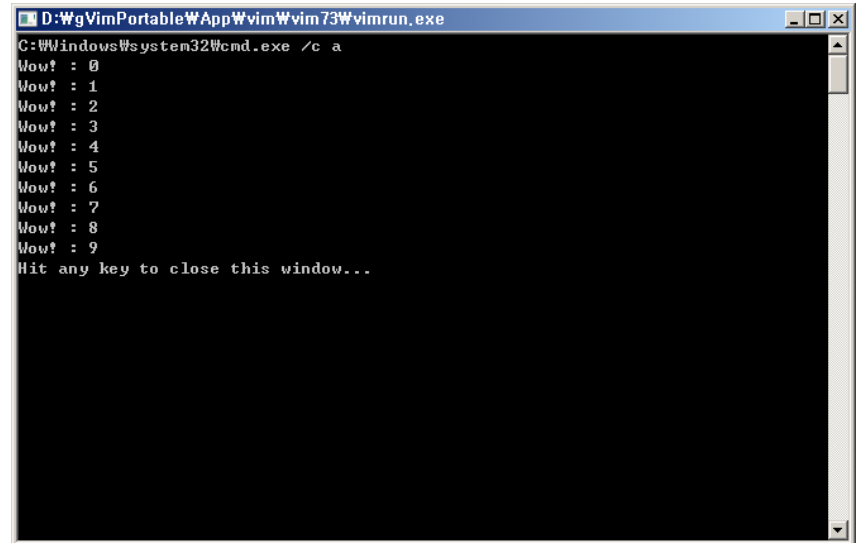


For Stmt.

▶ for

- ▶ 제일 중요함
- ▶ 마찬가지로 조건이 참일 때 반복, 거짓이면 종료

```
#include <stdio.h>
int main(){
    int i;
    for(i=0;i<10;i++){
        printf("Wow! : %d\n", i);
    }
    return 0;
}
```



```
D:\WgVimPortable\WApp\vim\vim73\vimrun.exe
C:\Windows\system32\cmd.exe /c a
Wow! : 0
Wow! : 1
Wow! : 2
Wow! : 3
Wow! : 4
Wow! : 5
Wow! : 6
Wow! : 7
Wow! : 8
Wow! : 9
Hit any key to close this window...
```



For Stmt.

- ▶ 반복문은 글로 배운다고 되는게 아님
 - ▶ 다른사람이 반복문을 어떻게 활용했나?
 - ▶ 어떤 상황에서 반복문을 어떻게 사용할까? 의 고민
- ▶ 앞으로 저랑 이런 연습 많이 할겁니다 ☺



Data type with I/O

Function

Conditional statement

for statement

Pointer

Algorithm



The End

- ▶ 문법은 이게 끝
 - ▶ 정말로 여기 있는 것들만 알면 모든 프로그램 다 짤 수 있음
 - ▶ 앞으로 남은 것들은 좀 더 편하게 코딩 하기 위해서!
- ▶ 이제 직접 여러 가지 코딩을 해 볼 것입니다
 - ▶ Welcome to Hell !
 - ▶ 물론 여러분에게 무작정 시키는 것은 아니고 같이 할겁니다
 - ▶ 앞으로도 파이팅

