

# C언어 강의자료

문정욱

# C언어 깊이 알기 1

## 라벨문(Label-statement)와 점프문(Jump-statement)

- 점프문(Jump Statement)
  - 지정된 라벨문으로 실행 위치 이동
- 라벨문(Label Statement)
  - "identifier:"로 시작
- 바람직하지 못한 문장
  - 실제 프로그램에서는 사용 하지 말 것.

### 입출력 결과

```
a=1
a=2
a=3
a=4
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    int a;

    a=1;
    L1: printf("a=%d\n",a);
    a=a+1;
    if(a<5) goto L1;

    return 0;
}
```

Label-statement

Jump-statement

# 반복문 비교

## ■ while 문

*while-statement:*  
**while**( *expression* )  
    *statement*

입출력 결과

```
1 2 3 4
1 2 3 4
계속하려면 아무 키나 누르십시오 . . .
```

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

```
int main(void)
{
```

```
    int n;
```

```
    n=1;
```

```
L1: if(n >= 5) goto L2;
    printf("%d ",n);
```

```
    ++n;
```

```
    goto L1;
```

```
L2: printf("\n");
```

```
    n=1;
```

```
    while( n<5 ) {
        printf("%d ",n);
        ++n;
```

```
    }
```

```
    printf("\n");
```

```
    return 0;
```

```
}
```

# 반복문 비교

## ■ for 문

*for-statement:*

```
for( statement, expression, statement )  
    statement
```

입출력 결과

```
1 2 3 4  
1 2 3 4  
계속하려면 아무 키나 누르십시오 . . .
```

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

```
int main(void)  
{
```

```
    int n;
```

```
    n=1;
```

```
L1: if(n >= 5) goto L2;  
    printf("%d ",n);
```

```
    ++n;
```

```
    goto L1;
```

```
L2: printf("\n");
```

```
    for( n=1; n<5; ++n ) {  
        printf("%d ",n);
```

```
    }
```

```
    printf("\n");
```

```
    return 0;
```

```
}
```

# 반복문 비교

## ■ do-while 문

*do-while-statement:*  
**do**  
    *statement*  
**while**(*expression*);

입출력 결과

```
1 2 3 4
1 2 3 4
계속하려면 아무 키나 누르십시오 . . .
```

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

```
int main(void)
{
```

```
    int n;
```

```
    n=1;
```

```
L1: printf("%d ",n);
```

```
    ++n;
```

```
    if(n<5) goto L1;
```

```
    printf("\n");
```

```
    n=1;
```

```
    do {
```

```
        printf("%d ",n);
```

```
        ++n;
```

```
    } while( n<5 );
```

```
    printf("\n");
```

```
    return 0;
```

```
}
```

# 반복문에서 break문

## ■ break-statement

- 반복문 안에서 break문은 반복문을 탈출시킨다.
- 반복문의 흐름을 복잡하게 만들 수 있으므로 사용을 자제해야 한다.

### 입출력 결과

```
i=0
i=1
i=2
i=3
i=4
i=
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    int i;

    for(i=0; i<100; ++i) {
        printf("i=");
        if(i==5)
            break;
        printf("%d\n",i);
    }
    printf("\n");
    return 0;
}
```

# 반복문에서 break문

## ■ break-statement

- break문은 가장 가까운 반복문만을 탈출한다.

### 입출력 결과

```
i=0
i=1
i=2
i=3
i=4
i=
i=0
i=1
i=2
i=3
i=4
i=
```

계속하려면 아무 키나 누르십시오 . . .

```
#include <stdio.h>

int main(void)
{
    int i,j;

    for(j=0; j<2; ++j) {
        for(i=0; i<100; ++i) {
            printf("i=");
            if(i==5)
                break;
            printf("%d\n",i);
        }
        printf("\n");
    }
    printf("\n");
    return 0;
}
```



# 반복문에서 continue문

## ■ continue-statement

- 처리해야 할 나머지 문장의 수행을 중지하고 반복문의 처음으로 이동한다.
- 반복문의 흐름을 복잡하게 만들 수 있으므로 사용을 자제해야 한다.

### 입출력 결과

```
i=0
i=1
i=2
i=3
i=4
i=i=98
i=99
계속하려면 아무 키나 누르십시오 . . .
```

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

```
int main(void)
```

```
{
```

```
    int i;
```

```
    for(i=0; i<100; ++i) {
```

```
        printf("i=");
```

```
        if(i==5) {
```

```
            i=97;
```

```
            continue;
```

```
        }
```

```
        printf("%d\n",i);
```

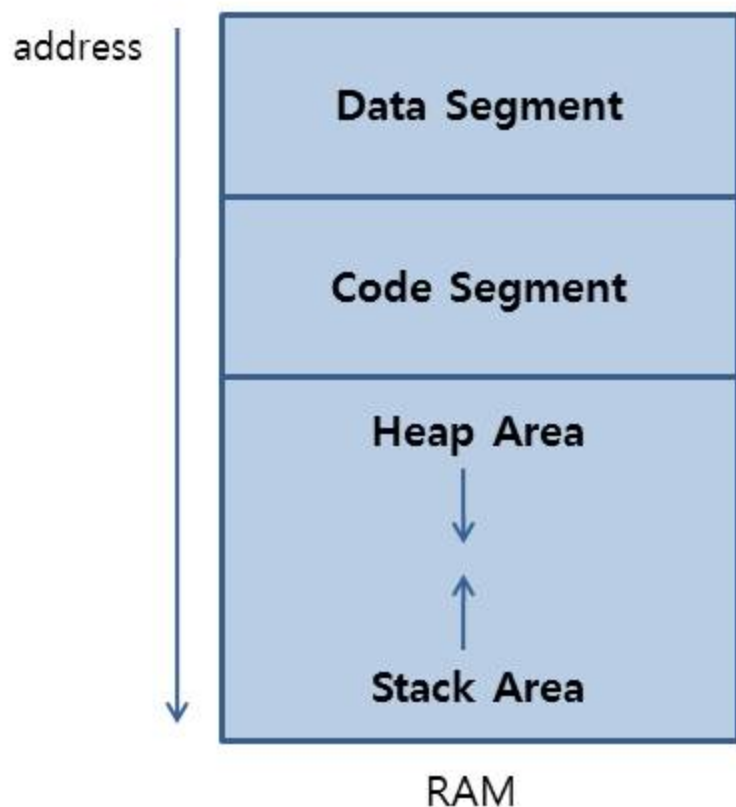
```
    }
```

```
    return 0;
```

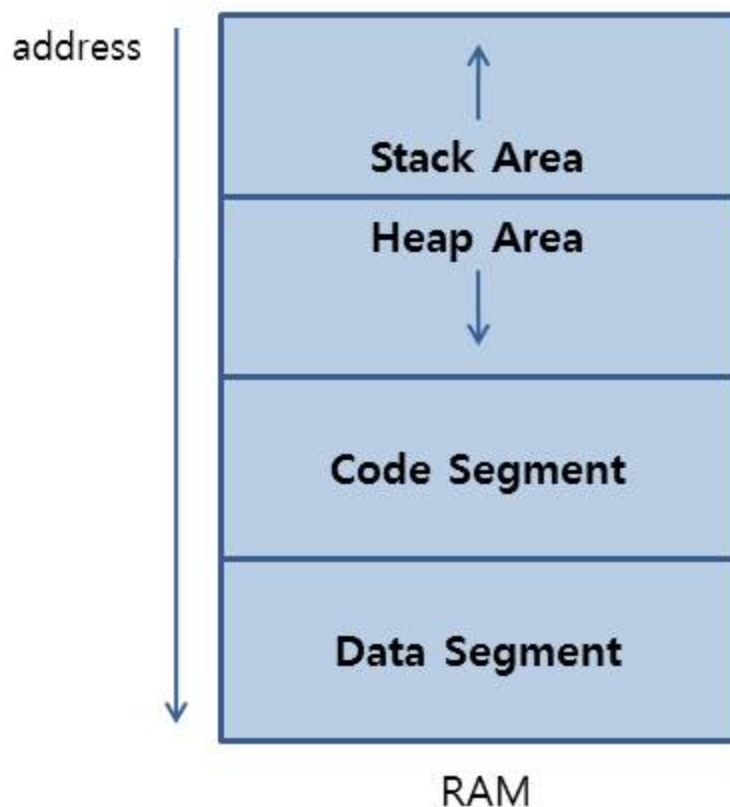
```
}
```

# 프로세스 구조(Process Structure)

## 일반적인 프로세스 구조



## MS Windows 프로세스 구조



# 메모리 영역별 변수 및 상수

- Data Segment
  - global variables, file-scope variables, block-scope variables
  - string constants
- Code Segment
  - executable machine codes
  - numerical constants
- Stack Area
  - local variables (in a block)
  - parameters (of a function)  $\subset$  local variables
- Heap Area
  - allocated memory segments

# 저장 위치에 따른 변수(variable)의 분류

Memory Area	Variables	Scope	Access
Data Segment	<ul style="list-style-type: none"><li>• global variables</li><li>• file-scope variables</li><li>• block-scope variables</li></ul>	<ul style="list-style-type: none"><li>[ declaration, EOF ]</li><li>[ declaration, EOF ]</li><li>[ declaration, EOB ]</li></ul>	<ul style="list-style-type: none"><li>• in all files</li><li>• in the corresponding file</li><li>• in the corresponding block</li></ul>
Stack	<ul style="list-style-type: none"><li>• local variables (in a block or a function)</li><li>• parameters <math>\subset</math> local variable (of a function)</li></ul>	<ul style="list-style-type: none"><li>[ declaration, EOB ]</li><li>[ declaration, EOB ]</li></ul>	<ul style="list-style-type: none"><li>• in the corresponding block</li><li>• in the corresponding function</li></ul>

# 구조체(struct)

- 자기 참조 구조체
  - 포인터를 사용하여 자신과 동일한 구조체 공간을 가리킬 수 있다.

```
#include <stdio.h>

struct node {
    int a;
    struct node* p;
};

int main(void)
{
    struct node n1; // ok
    node n2;        // error

    return 0;
}
```

# 구조체(struct)

- 자기 참조 구조체
  - typedef의 활용

```
#include <stdio.h>

typedef struct tag_node node;
struct tag_node {
    int a;
    node* p;
};

int main(void)
{
    struct tag_node n1; // ok
    node n2;             // ok

    return 0;
}
```

# 구조체(struct)

## 자기 참조 구조체 사용 예(Linked List)

```
#include <stdio.h>

struct tag_node {
    int a;
    struct tag_node* p;
};

void f(struct tag_node* ps)
{
    while(ps) {
        printf("%d\n", ps->a);
        ps=ps->p;
    }
}
```

```
int main(void)
{
    struct tag_node s1={1,NULL};
    struct tag_node s2={2,NULL};
    struct tag_node s3={3,NULL};

    s1.p=&s2;
    s2.p=&s3;
    s3.p=NULL;

    f(&s1);
    return 0;
}
```

입출력 결과

```
1
2
3
계속하려면 아무 키나 누르십시오 . . .
```



# 구조체(struct)

## 자기 참조 구조체 선언 1

```
#include <stdio.h>

struct node {
    int data;
    struct node* next;
};

int main(void)
{
    struct node n3={3,NULL};
    struct node n2={2,&n3};
    struct node n1={1,&n2};
    struct node* head=&n1;
    struct node* p;

    for(p=head;p!=NULL;p=p->next)
        printf("%p: (%d, %p)\n",
            p, p->data, p->next);
    return 0;
}
```

## 자기 참조 구조체 선언 2

```
#include <stdio.h>

struct tag_node {
    int data;
    struct tag_node* next;
};
typedef struct tag_node node;

int main(void)
{
    node n3={3,NULL};
    node n2={2,&n3};
    node n1={1,&n2};
    node* head=&n1;
    node* p;

    for(p=head;p!=NULL;p=p->next)
        printf("%p: (%d, %p)\n",
            p, p->data, p->next);
    return 0;
}
```



# 구조체(struct)

## 자기 참조 구조체 선언 3

```
#include <stdio.h>

typedef struct tag_node node;
struct tag_node {
    int data;
    node* next;
};

int main(void)
{
    node n3={3,NULL};
    node n2={2,&n3};
    node n1={1,&n2};
    node* head=&n1;
    node* p;

    for(p=head;p!=NULL;p=p->next)
        printf("%p: (%d, %p)\n",
            p, p->data, p->next);
    return 0;
}
```

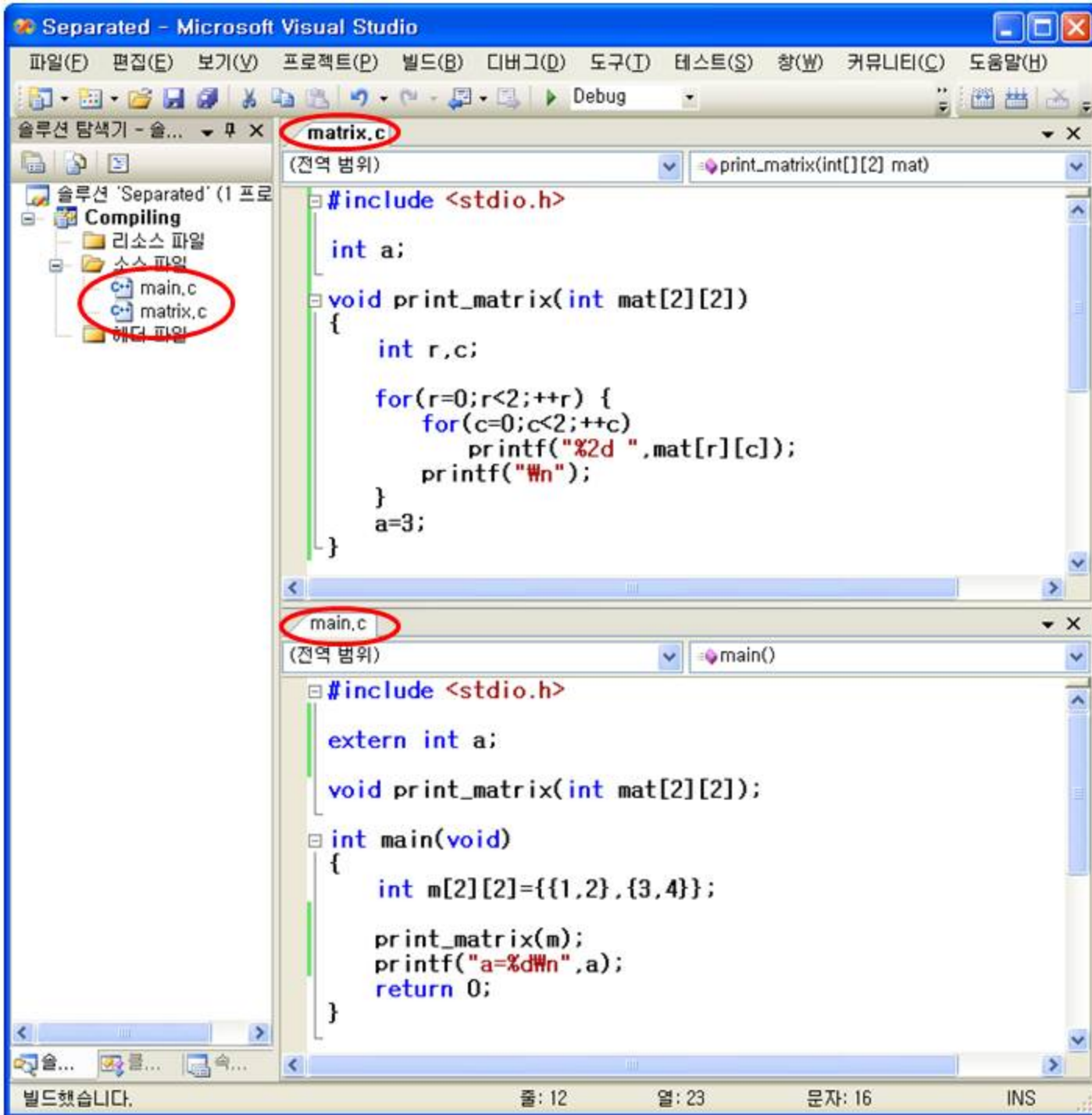
## 자기 참조 구조체 선언 4

```
#include <stdio.h>

typedef struct tag_node {
    int data;
    struct tag_node* next;
} node;

int main(void)
{
    node n3={3,NULL};
    node n2={2,&n3};
    node n1={1,&n2};
    node* head=&n1;
    node* p;

    for(p=head;p!=NULL;p=p->next)
        printf("%p: (%d, %p)\n",
            p, p->data, p->next);
    return 0;
}
```



# 분할 컴파일(separated compile)

## main.c

```
#include <stdio.h>

extern int a;  변수 참조 선언

void print_matrix(int mat[2][2]);  함수 참조 선언

int main(void)
{
    int m[2][2]={{1,2},{3,4}};

    print_matrix(m);
    printf("a=%d\n",a);
    return 0;
}
```

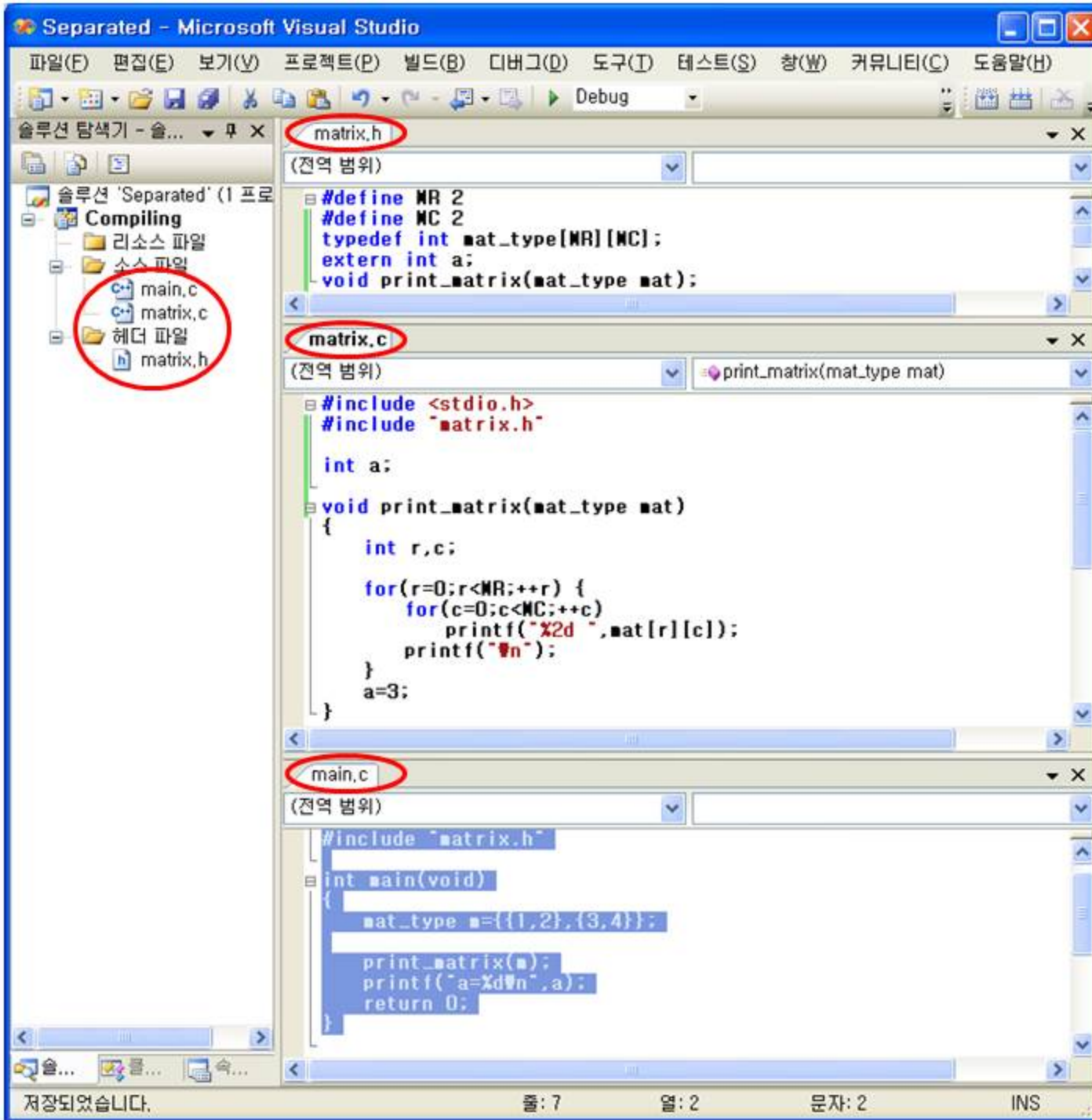
## matrix.c

```
#include <stdio.h>

int a;  외부 변수 선언

void print_matrix(int mat[2][2])  함수 정의
{
    int r,c;

    for(r=0;r<2;++r) {
        for(c=0;c<2;++c)
            printf("%2d ",mat[r][c]);
        printf("\n");
    }
    a=3;
}
```



# 분할 컴파일(separated compile)

## matrix.h

매크로 및 타입 정의

```
#define MR 2
#define MC 2

typedef int mat_type[MR][MC];

extern int a;
void print_matrix(mat_type mat);
```

## matrix.c

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

int a;

void print_matrix(mat_type mat)
{
    int r,c;

    for(r=0;r<MR;++r) {
        for(c=0;c<MC;++c)
            printf("%2d ",mat[r][c]);
        printf("\n");
    }
    a=3;
}
```



# 분할 컴파일(separated compile)

## matrix.h

매크로 및 타입 정의

```
#define MR 2  
#define MC 2  
  
typedef int mat_type[MR][MC];
```

```
extern int a;  
void print_matrix(mat_type mat);
```

변수 및 함수의 참조 선언

## main.c

```
#include <stdio.h>  
#include "matrix.h"  
  
int main(void)  
{  
    mat_type m={{1,2},{3,4}};  
  
    print_matrix(m);  
    printf("a=%d\n",a);  
    return 0;  
}
```

# 정적(static) 전역 변수

## ■ 정적(static) 변수

- 분할 컴파일을 할 경우 변수가 선언된 파일 내에서만 참조가 가능한 전역 변수

변수 종류	생성 및 파괴	참조 범위
지역 변수	블록의 시작 및 종료	선언된 위치에서 블록의 끝까지
전역 변수	프로그램의 시작 및 종료	선언된 위치에서 프로그램의 끝까지
정적(static) 지역 변수	프로그램의 시작 및 종료	선언된 위치에서 블록의 끝까지
정적(static) 전역 변수	프로그램의 시작 및 종료	선언된 위치에서 파일의 끝까지

입출력 결과

```

func: 4
main: 3
계속하려면 아무 키나 누르십시오 . . .
  
```

```

// funct.c
#include <stdio.h>
  
```

이름은 같지만 서로 다른 변수  
선언된 파일에서만 사용가능

```

static int a;

void funct(void)
{
    a = 4;
    printf("func: %d\n", a);
}
  
```

```

// main.c
#include <stdio.h>
  
```

```

void funct(void);

static int a = 3;

int main(void)
{
    funct();
    printf("main: %d\n", a);
    return 0;
}
  
```

# 정적(static) 전역 변수

## 2개의 정적 전역 변수의 사용

```
// funct.c
#include <stdio.h>

static int a;

void funct(void)
{
    a = 4;
    printf("funct: %d\n", a);
}
```

```
// main.c
#include <stdio.h>

void funct(void);

static int a = 3;

int main(void)
{
    funct();
    printf("main: %d\n", a);
    return 0;
}
```

입출력 결과

```
funct: 4
main: 3
계속하려면 아무 키나
누르십시오 . . .
```

## 1개의 전역 변수의 사용

```
// funct.c
#include <stdio.h>

extern int a;

void funct(void)
{
    a = 4;
    printf("funct: %d\n", a);
}
```

main.c의 전역 변수 a를  
참조하기 위한 참조 선언

```
// main.c
#include <stdio.h>

void funct(void);

int a = 3;

int main(void)
{
    funct();
    printf("main: %d\n", a);
    return 0;
}
```

입출력 결과

```
funct: 4
main: 4
계속하려면 아무 키나
누르십시오 . . .
```



# 정적(static) 전역 변수

## 2개의 정적 전역 변수의 사용

```
// funct.c
#include <stdio.h>

static int a;

void funct(void)
{
    a = 4;
    printf("funct: %d\n", a);
}
```

```
// main.c
#include <stdio.h>

void funct(void);

static int a = 3;

int main(void)
{
    funct();
    printf("main: %d\n", a);
    return 0;
}
```

## 1개의 정적 전역 변수의 사용

```
// funct.c
#include <stdio.h>

extern int a;

void funct(void)
{
    a = 4;
    printf("funct: %d\n", a);
}
```

main.c의 전역 변수 a를  
참조하기 위한 참조 선언

```
// main.c
#include <stdio.h>

void funct(void);

static int a = 3;

int main(void)
{
    funct();
    printf("main: %d\n", a);
    return 0;
}
```

main.c 이외의 파일에서는  
참조가 불가능.

**Syntax Error  
(Link Error)**

# 저장 위치에 따른 변수(variable)의 분류

```
#include <stdio.h>

int a=1;           // (1) global
static int b=2;    // (2) file-scope

void f(int a,int b) // (3,4) local
{
    int c=6;        // (6) local
    static int d=7; // (7) block-scope
    printf("a3=%d b4=%d ",a,b);
    printf("c6=%d d7=%d\n",c,d);
    d++;
    {
        int c=8;    // (8) local
        printf("c8=%d\n",c);
    }
    printf("c=%d\n",c);
}
```

```
int main(void)
{
    static int c=9; // (9) block-scope
    int a=10;       // (10) local

    printf("a10=%d c9=%d\n",a,c);
    f(3,4);
    f(3,4);
    return 0;
}
```

## 입출력 결과

```
a10=10 c9=9
a3=3 b4=4 c6=6 d7=7
c8=8
c=6
a3=3 b4=4 c6=6 d7=8
c8=8
c=6
계속하려면 아무 키나 누르십시오 . . .
```

# Floating-point Constant

- 과학적 표현(Scientific Notation)

$1.575\text{E}-2 == 1.575 \times 10^{-2}$

$1.575\text{e}-2 == 1.575 \times 10^{-2}$

입출력 결과

15.750000

15.750000

15.750000

-0.002500

0.002500

계속하려면 아무 키나 누르십시오 . . .

```
#include <stdio.h>

int main(void)
{
    double d1=15.75;
    double d2=1.575E1;    // 15.75
    double d3=1575e-2;    // 15.75
    double d4=-2.5e-3;    // -0.0025
    double d5=25E-4;      // 0.0025

    printf("%f\n",d1);
    printf("%f\n",d2);
    printf("%f\n",d3);
    printf("%f\n",d4);
    printf("%f\n",d5);
    return 0;
}
```

# Floating-point Constant

## ■ Constant Type

Suffix	Type
F, f	float
(none)	double
L, l	long double

```
#include <stdio.h>

int main(void)
{
    float d1=3.14F;
    float d2=3.14f;
    double d3=3.14;
    long double d4=3.14L;
    long double d5=3.14l;

    return 0;
}
```

# Integer Constant

## ■ Representation

Base	Representation
Octal	0~
Decimal	~
Hexadecimal	0X~

### 입출력 결과

```
347 10 32179
533 12 76663
15b a 7db3
15B A 7DB3
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    int a1=347; // Decimal
    int a2=012; // Octal
    int a3=0x7dB3; // Hexadecimal

    printf("%d %d %d\n",a1,a2,a3);
    printf("%o %o %o\n",a1,a2,a3);
    printf("%x %x %x\n",a1,a2,a3);
    printf("%X %X %X\n",a1,a2,a3);

    return 0;
}
```

# Integer Constant

## ■ Constant Type

Suffix	Type
U, u	unsigned
L, l	long
i64	long long

```
#include <stdio.h>

int main(void)
{
    long a1=79L;
    long a2=012L;
    long a3=0xaL;
    unsigned int a4=23u;
    unsigned long a5=77UL;

    return 0;
}
```



# Character Constant

## ■ Constant Type

Prefix	Type
(none)	char
L	wchar_t

입출력 결과

```
1
2
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

int main(void)
{
    char ch = 'a';      // ascii
    wchar_t ch2 = L'a'; // unicode

    printf("%d\n", sizeof ch);
    printf("%d\n", sizeof ch2);
    return 0;
}
```

# Character Constant

## ■ Escape Sequence

Esc. Seq.	Meaning
<code>\a</code>	Bell (alert)
<code>\b</code>	Backspace
<code>\f</code>	Formfeed
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Horizontal tab
<code>\v</code>	Vertical tab
<code>\'</code>	Single quotation mark
<code>\"</code>	Double quotation mark
<code>\\</code>	Backslash
<code>\?</code>	Literal question mark
<code>\ooo</code>	ASCII character in octal notation
<code>\xhh</code>	ASCII character in hexadecimal notation
<code>\xhhhh</code>	Unicode character in hexadecimal notation

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

```
int main(void)
```

```
{
```

```
    char ch1='\b';
```

```
    char ch2='\t';
```

```
    char ch3='\x61'; // 'a'
```

```
    char ch4='\141'; // 'a'
```

```
    printf("123%c%c123\n",ch1,ch2);
```

```
    printf("\"\"?\n");
```

```
    printf("%c %c\n",ch3,ch4);
```

```
    return 0;
```

```
}
```

입출력 결과

```
12      123
```

```
"?"
```

```
a a
```

```
계속하려면 아무 키나 누르십시오 . . .
```



# Character Constant

- ASCII Code
  - American Standard Code for Information Interchange Code

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

# 중첩 구조체(union)

## ■ 특징

- 메모리를 중첩해서 할당하므로, 실제 사용하게 되는 변수는 1개이다.
- 타입의 크기는 요소 중에서 가장 큰 타입의 크기가 된다.

입출력 결과

8

계속하려면 아무 키나 누르십시오 . . .

```
#include <stdio.h>

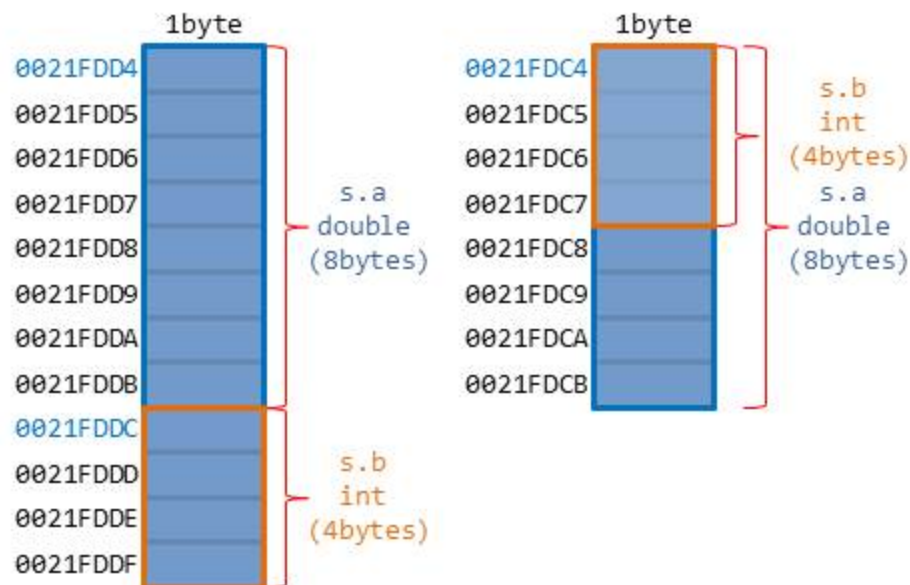
union tag_un {
    double a;
    int b;
};

int main(void)
{
    union tag_un u;

    printf("%d\n", sizeof u);
    return 0;
}
```

# 중첩 구조체(union) vs 구조체(struct)

## ■ 구조 비교



### 입출력 결과

```
0040FC90 0040FC90 0040FC98
0040FC80 0040FC80 0040FC80
계속하려면 아무 키나 누르십시오 . . .
```

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

```
struct tag_st {
    double a;
    int b;
};
```

```
union tag_un {
    double a;
    int b;
};
```

```
int main(void)
{
    struct tag_st s;
    union tag_un u;

    printf("%p %p %p\n", &s, &s.a, &s.b);
    printf("%p %p %p\n", &u, &u.a, &u.b);
    return 0;
}
```

# Const Variable

## ■ 특징

- 값이 변경될 수 없는 변수
- 변수 선언시 초기화해야 한다.
- 한번 초기화되면 변경할 수 없다.
- 함수 인자로 쓰일 때는 인자로 전달받은 값으로 초기화되고
- 초기화된 후 변경할 수 없다.

```
#include <stdio.h>

void f(const int v)
{
    v=3; /* error */
    printf("%d\n",v);
}

int main(void)
{
    const int a=3;

    a=4; /* error */
    f(a);
    return 0;
}
```

# Const Variable

- 포인터의 타입 변환
  - 참조 타입이 const가 아닐 경우 const 형으로 변환이 가능하다.
  - 참조 타입이 const일 경우 const가 아닌 타입으로 변환이 바람직하지 못 하다. (C언어에서는 경고 처리, C++에서는 예러처리)

```
#include <stdio.h>

int main(void)
{
    const int a=3;
    int b;

    const int* p;
    int* q;

    p=&a; // const int* -> const int*
    q=&b; // int* -> int*

    p=&b; // int* -> const int*
    q=&a; // const int* -> int* : warning
    return 0;
}
```

# 열거형(enum)

## ■ 특징

- 사용자 정의 타입
- 실제로는 정수형
- 실제 값은 0부터 1씩 증가
- 실제 값을 직접 설정가능

입출력 결과

0 10 11

계속하려면 아무 키나 누르십시오 . . .

```
#include <stdio.h>

enum fruit {
    apple, strawberry,
    pear=10, banana
};

int main(void)
{
    enum fruit f1,f2,f3;

    f1=apple;
    f2=pear;
    f3=banana;
    printf("%d %d %d\n",f1,f2,f3);
    return 0;
}
```



# 가변 인자 목록(Variable-argument Lists)

```
#include <stdio.h>

void f(int first, int a, int b, int c)
{
    printf("%p: %d\n", &a, a);
    printf("%p: %d\n", &b, b);
    printf("%p: %d\n", &c, c);
}

int main(void)
{
    f(99,1,2,3);
    return 0;
}
```

## 입출력 결과

```
0015FC54: 1
0015FC58: 2
0015FC5C: 3
계속하려면 아무 키나 누르십시오 . . .
```

## 입출력 결과

```
001DF674: 1
001DF678: 2
001DF67C: 3
계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>

void f(int first, ... )
{
    char* p;

    p=(char*)&first + sizeof(int);

    printf("%p: ", p);
    p+=sizeof(int);
    printf("%d\n", *(int*)(p-sizeof(int)) );

    printf("%p: ", p);
    p+=sizeof(int);
    printf("%d\n", *(int*)(p-sizeof(int)) );

    printf("%p: ", p);
    p+=sizeof(int);
    printf("%d\n", *(int*)(p-sizeof(int)) );

    p=NULL;
}

int main(void)
{
    f (99,1,2,3);
    return 0;
}
```

# 가변 인자 목록(Variable-argument Lists)

```
#include <stdio.h>

void f (int first, ... )
{
    char* p;

    p=(char*)&first + sizeof(int);

    printf("%p: ", p);
    printf("%d\n",
        *(int*)((p+=sizeof(int))-sizeof(int)) );

    printf("%p: ", p);
    printf("%d\n",
        *(int*)((p+=sizeof(int))-sizeof(int)) );

    printf("%p: ", p);
    printf("%d\n",
        *(int*)((p+=sizeof(int))-sizeof(int)) );

    p=NULL;
}
```

## 입출력 결과

```
0036F950: 1
0036F954: 2
0036F958: 3
계속하려면 아무 키나 누르십시오 . . .
```

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

void f(int first, ... )
{
    va_list p;

    va_start(p,first);

    printf("%p: ", p);
    printf("%d\n", va_arg(p,int) );

    printf("%p: ", p);
    printf("%d\n", va_arg(p,int) );

    printf("%p: ", p);
    printf("%d\n", va_arg(p,int) );

    va_end(p);
}
```

## 입출력 결과

```
0036FADC: 1
0036FAE0: 2
0036FAE4: 3
계속하려면 아무 키나 누르십시오 . . .
```



# 가변 인자 목록(Variable-argument Lists)

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

void print(int first, ... )
{
    va_list p;

    va_start(p, first);
    printf("%p: ", p);
    printf("%c\n", va_arg(p, char) );
    printf("%p: ", p);
    printf("%hd\n", va_arg(p, short) );
    printf("%p: ", p);
    printf("%d\n", va_arg(p, int) );
    printf("%p: ", p);
    printf("%ld\n", va_arg(p, long) );
    printf("%p: ", p);
    printf("%lld\n", va_arg(p, long long) );
    printf("%p: ", p);
    printf("%f\n", va_arg(p, double) );
    printf("%p: ", p);
    printf("%f\n", va_arg(p, double) );
    printf("%p: ", p);
    printf("%f\n", va_arg(p, double) );
    va_end(p);
}
```

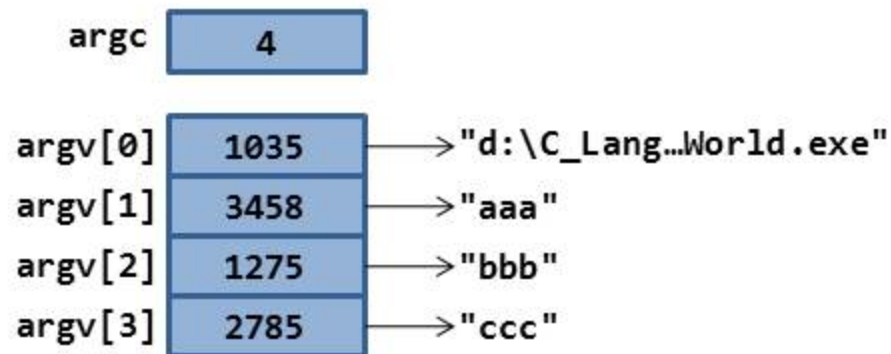
```
int main(void)
{
    char c='a';
    short s=2;
    int i=3;
    long l=4L;
    long long ll=5L;
    float f=3.14f;
    double d=3.14;
    long double ld=3.14L;
    print(9999,c,s,i,l,ll,f,d,ld);
    return 0;
}
```

## 입출력 결과

```
0046F7CC: a
0046F7D0: 2
0046F7D4: 3
0046F7D8: 4
0046F7DC: 5
0046F7E4: 3.140000
0046F7EC: 3.140000
0046F7F4: 3.140000
계속하려면 아무 키나 누르십시오 . . .
```

# Program Parameter

## ■ argv[]의 구조



```
#include <stdio.h>

int main(int argc, char* argv[])
{
    int i;

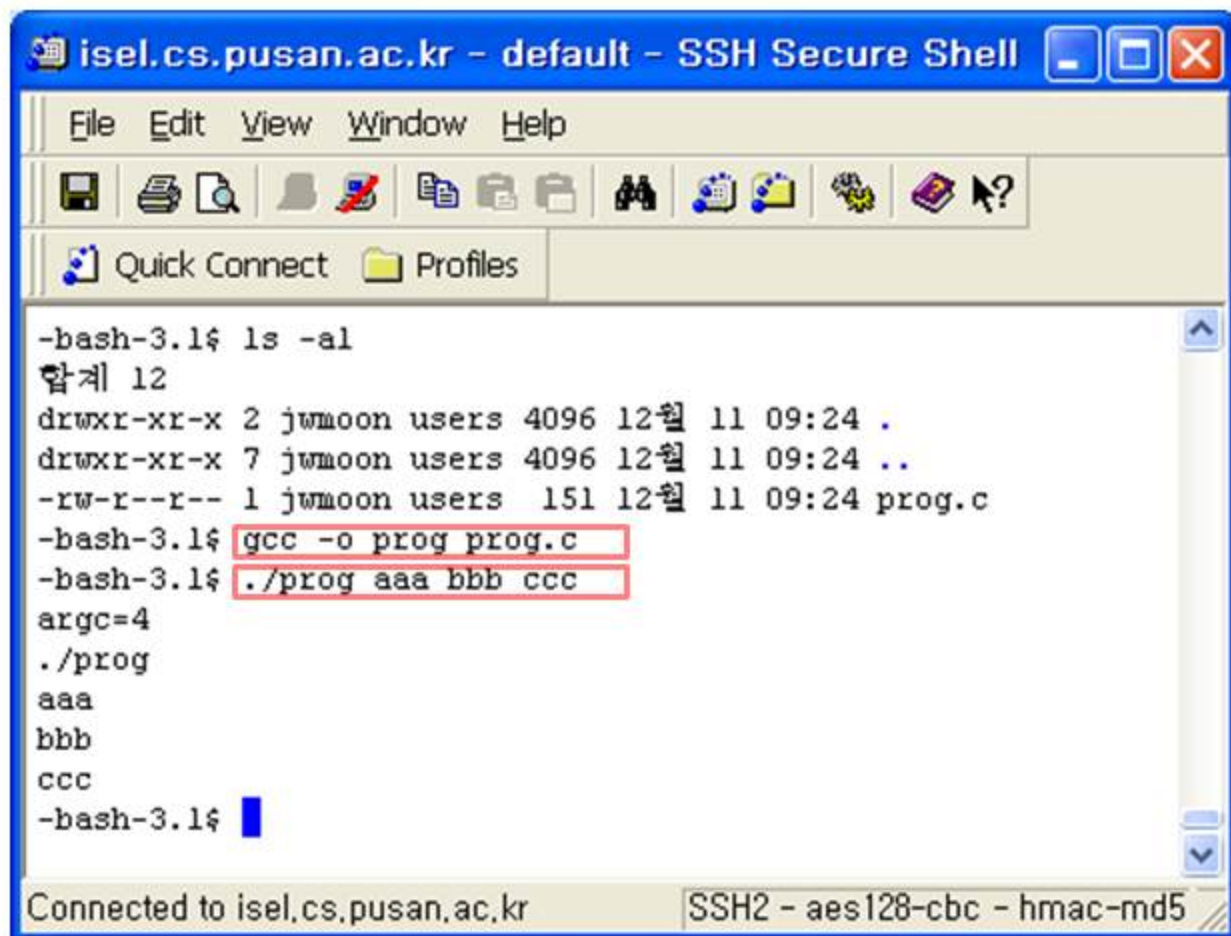
    printf("argc=%d\n", argc);
    for(i=0; i<argc; ++i)
        printf("%s\n", argv[i]);

    return 0;
}
```

### 입출력 결과

```
argc=4
c:\clang\Debug\cprog.exe
aaa
bbb
ccc
계속하려면 아무 키나 누르십시오 . . .
```

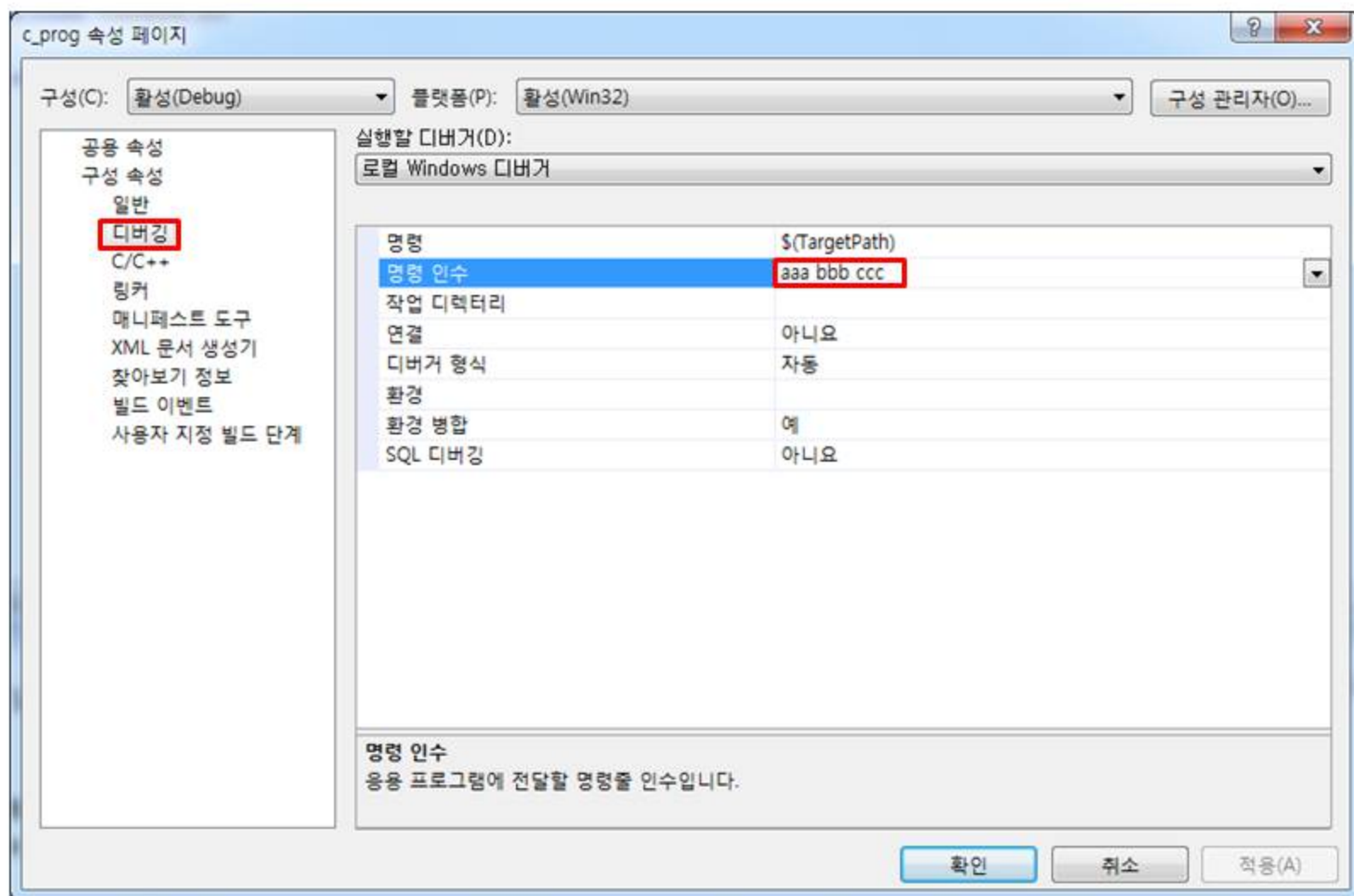
# LINUX 환경에서 컴파일 및 실행



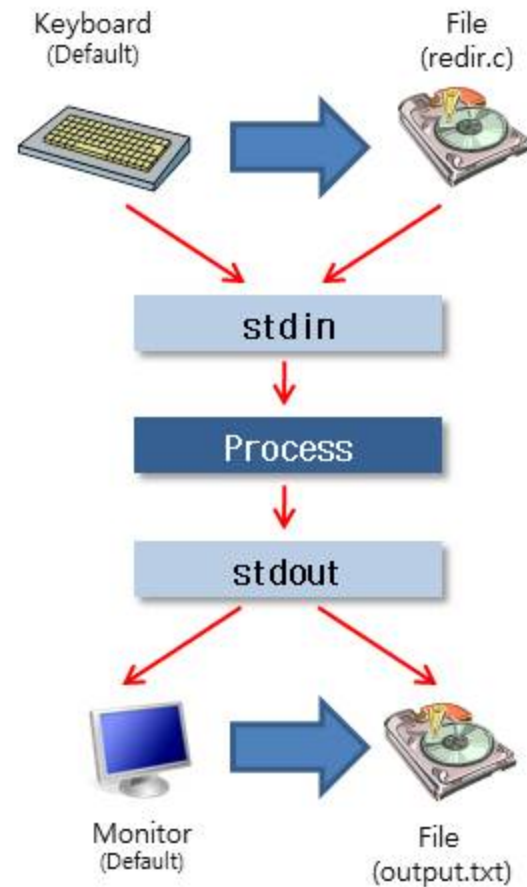
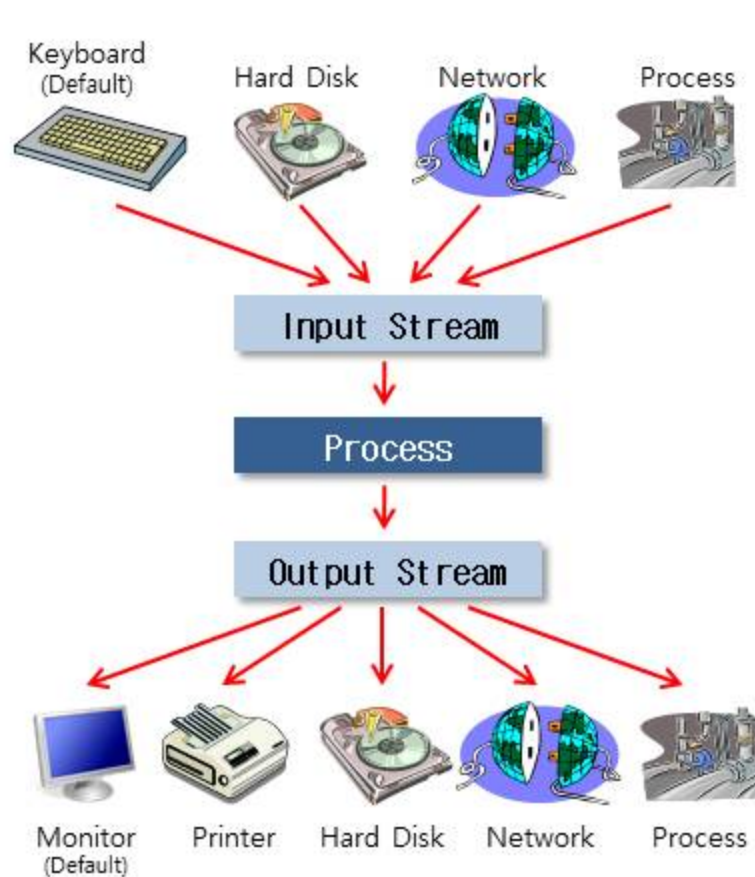
The screenshot shows an SSH terminal window titled "isel.cs.pusan.ac.kr - default - SSH Secure Shell". The window has a menu bar (File, Edit, View, Window, Help) and a toolbar with various icons. Below the toolbar are tabs for "Quick Connect" and "Profiles". The terminal content shows a user running the command `ls -al`, which lists the contents of the current directory. The output shows a directory listing with permissions, owner, group, size, date, and filename. The user then runs `gcc -o prog prog.c` to compile the program, followed by `./prog aaa bbb ccc` to execute it. The output of the program is `argc=4`, `./prog`, `aaa`, `bbb`, and `ccc`. The terminal status bar at the bottom indicates "Connected to isel.cs.pusan.ac.kr" and "SSH2 - aes128-cbc - hmac-md5".

```
isel.cs.pusan.ac.kr - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
-bash-3.1$ ls -al
합계 12
drwxr-xr-x 2 jwmoon users 4096 12월 11 09:24 .
drwxr-xr-x 7 jwmoon users 4096 12월 11 09:24 ..
-rw-r--r-- 1 jwmoon users 151 12월 11 09:24 prog.c
-bash-3.1$ gcc -o prog prog.c
-bash-3.1$ ./prog aaa bbb ccc
argc=4
./prog
aaa
bbb
ccc
-bash-3.1$
Connected to isel.cs.pusan.ac.kr SSH2 - aes128-cbc - hmac-md5
```

# MS Visual Studio에서 프로그램 인자 전달

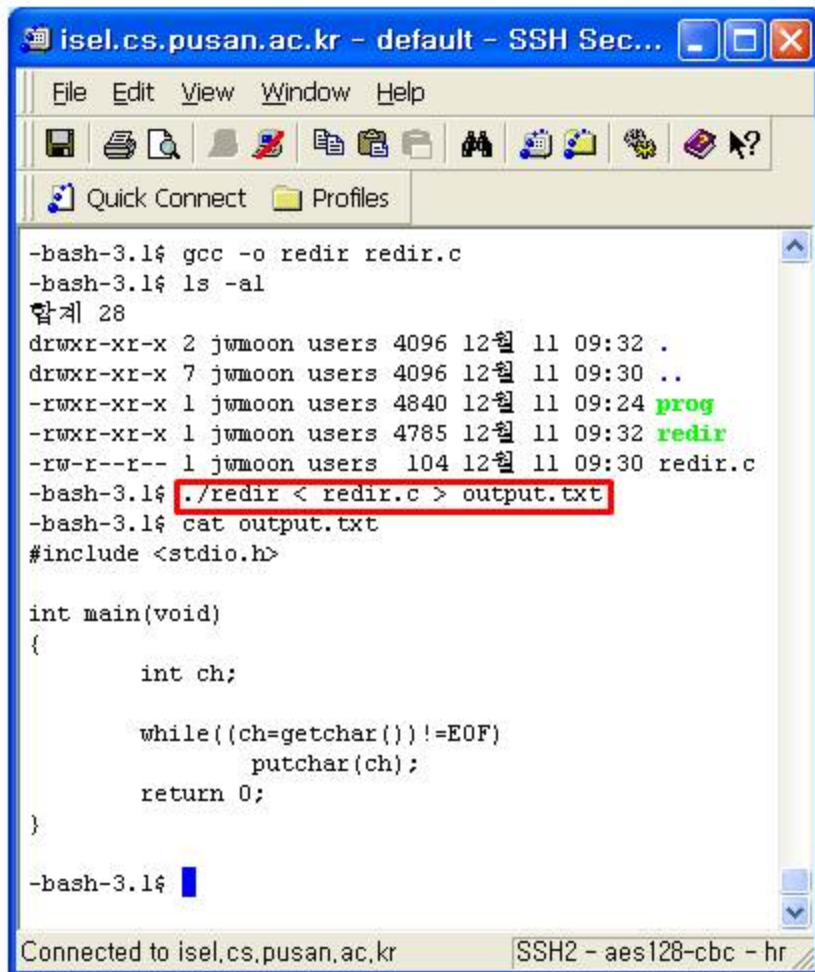


# Redirection





# Redirection



The screenshot shows an SSH terminal window titled "isel.cs.pusan.ac.kr - default - SSH Sec...". The terminal displays the following commands and output:

```
-bash-3.1$ gcc -o redir redir.c
-bash-3.1$ ls -al
합계 28
drwxr-xr-x 2 jwmoon users 4096 12월 11 09:32 .
drwxr-xr-x 7 jwmoon users 4096 12월 11 09:30 ..
-rwxr-xr-x 1 jwmoon users 4840 12월 11 09:24 prog
-rwxr-xr-x 1 jwmoon users 4785 12월 11 09:32 redir
-rw-r--r-- 1 jwmoon users 104 12월 11 09:30 redir.c
-bash-3.1$ ./redir < redir.c > output.txt
-bash-3.1$ cat output.txt
#include <stdio.h>

int main(void)
{
    int ch;

    while((ch=getchar())!=EOF)
        putchar(ch);

    return 0;
}

-bash-3.1$
```

The command `./redir < redir.c > output.txt` is highlighted with a red box. The output of the program is shown in the `cat output.txt` command, displaying the source code of `redir.c`.

```
#include <stdio.h>

int main(void)
{
    int ch;

    while((ch=getchar())!=EOF)
        putchar(ch);

    return 0;
}
```