

CS SCHOOL

변수란 무엇인가?

- 변수와 메모리

```
int num = 15;
```

(이 한 문장 안에 엄청난 의미가 담겨있다)

상수도 스택 영역에 할당된 후 다음 행에서 소멸
호출할 수단이 없으므로

'같다'가 아닌 대입

이름이 없다면 다음 행에서 사라집니다

1. 변수 선언 위치에 따라 메모리 저장 위치와 생성 소멸 시기가 결정

```
1  #include <stdio.h>
2
3  int num = 5; // 전역변수
4
5  void Func(int n) // 매개변수
6  {
7      int num = 5; // 지역변수
8  }
```

→ data 영역

→ stack 영역

→ stack 영역

변수란 무엇인가?

- 변수의 선언과 초기화

```
int num1;  
num1 = 5;
```

변수의 선언

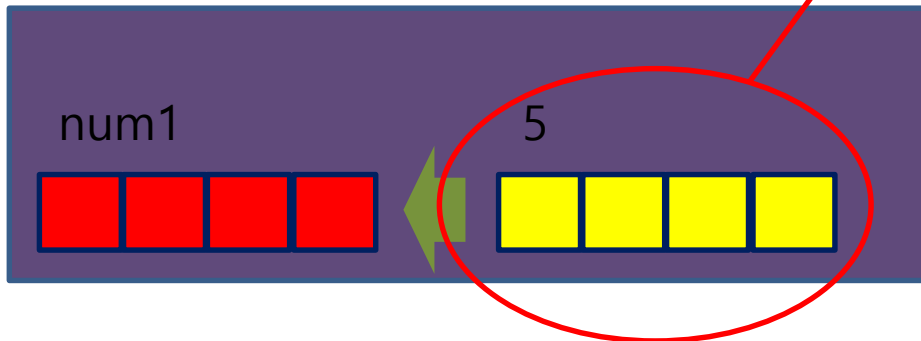
초기화 5도 메모리에 할당

이름이 없어 접근이 불가능하므로 다음 라인에서 해제.

```
int num2 = 10;
```

변수 선언 및 초기화

STACK 영역



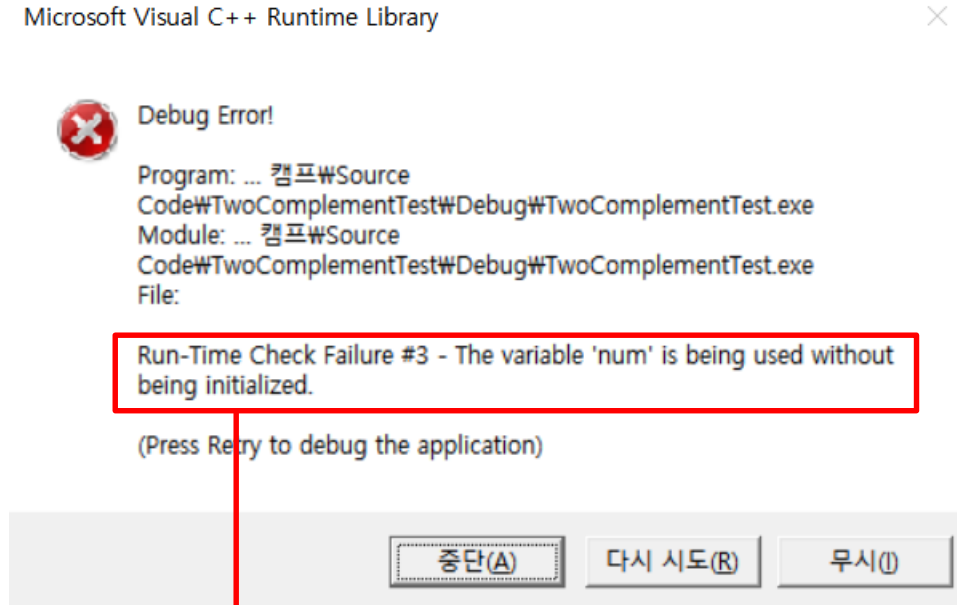
대입 연산 이후 해제

변수란 무엇인가?

- 변수의 선언과 초기화

```
int num;
```

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



변수를 초기화하지 않았다는 에러메시지

변수란 무엇인가?

- 변수의 이름이 되기 위한 조건

1. 알파벳, 숫자, 언더바(_) 조합
2. 대소문자 구분
3. 숫자로 시작할 수 없다
4. C언어 문법 구성 키워드 사용 불가
ex) enum, float, struct, sizeof
5. 공백 포함 불가

```
int num_ = 10;
```

```
int num77 = 10;
```

```
int NUM = 10;
```

대문자로 시작하는 변수는 잘 쓰지 않아요!

```
int num = 10;
```

```
int 77num = 10;
```

숫자가 앞에 옴

```
int enum = 10;
```

키워드 사용

```
int study num = 10;
```

공백 불가~

자료형

Data type의 종류

구분	자료형	범위	바이트
정수형	char	-128 ~ 127	1(8)
	unsigned char	0 ~ 255	1(8)
	short	-32768 ~ 32767	2(16)
	int	-2,147,483,648 ~ 2,147,483,647	4(32)
	long	-2,147,483,648 ~ 2,147,483,647	4(32)
	unsigned short	0~65535	2(16)
	unsigned int	0~4,294,967,295	4(32)
	unsigned long	0~4,294,967,295	4(32)
실수형	float	$8.4 \times 10^{-37} \sim 3.4 \times 10^{38}$	4(32)
	double	$2.2 \times 10^{-308} \sim 1.8 \times 10^{308}$	8(64)
나열형	enum	정수를 대신하여 사용하는 별명, int형의 크기	
무치형	void	실제 자료는 없음을 명시적으로 선언	

자료형

Data type의 종류

1. 수

- 1) 정수 양의 정수, 음의 정수
ex) 1, 13, -4, -46
- 2) 실수
ex) 3.13, -5.234

2. 문자

- 1) 문자 문자는 말 그대로 문자 하나!
ex) 'a', 'b'
- 2) 문자열 문자가 모인 것!!
ex) "I like the toy!"

자료형

정수를 나타내는 자료형

1. 정수

1) int

4 byte, 표현 범위 : -2,147,483,648 ~ 2,147,483,647

2) Short (거의 쓰이지 않고 소켓 프로그래밍에서만 쓰임)

2 byte, 표현 범위 : -32,768 ~ 32,767

3) char (문자 표현을 위해 탄생한 자료형, 하지만 어쨌든 정수 표현 가능)

1 byte, 표현 범위 : -128 ~ 127

자료형

실수를 나타내는 자료형

2. 실수 (표현 범위는 넓지만 정확도가 떨어집니다)

float? double? 선택 기준은?

메모리가 엄청난 이슈가 아니라면 무조건 double!!

따져야 한다면 가장 중요하게 볼 것은 정밀도!!

자료형	표현범위	소수점 이하 정밀도	바이트 수
float	10^{-37} 이상 10^{38} 이하	6자리	4
double	10^{-307} 이상 10^{308} 이하	15자리	8

자료형

문자를 나타내는 자료형

3. 문자

문자는 'A' : 작은 따옴표 안에 한 글자만을 문자라고 함.

문자를 위한 자료형 : char (character)

char ch = 'A';

자료형

- ASCII 코드

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	.
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					

가상메모리(Virtual Memory)

1. 가상메모리란?

어디서 확인할 수 있을지 궁금하신가요??

시스템->고급 시스템 설정->고급->성능-> 고급->가상메모리



물리적인 RAM



pagefile.sys

하드디스크

가상메모리

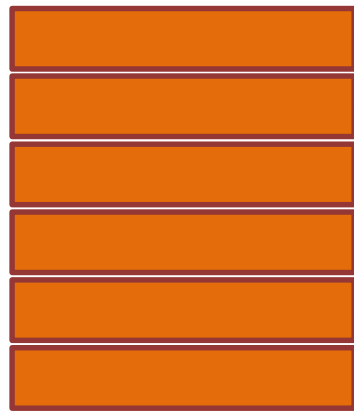
메모리(RAM)가 모자랄 경우,

- 안 쓰는 데이터를 하드디스크의 페이징 파일(Paging File)에 저장
- 다시 쓸 일이 생기면 swap을 통해 다시 RAM으로 가져온다.

가상메모리(Virtual Memory)

2. 페이지 테이블

페이지 크기 : 4096 bytes(4K)



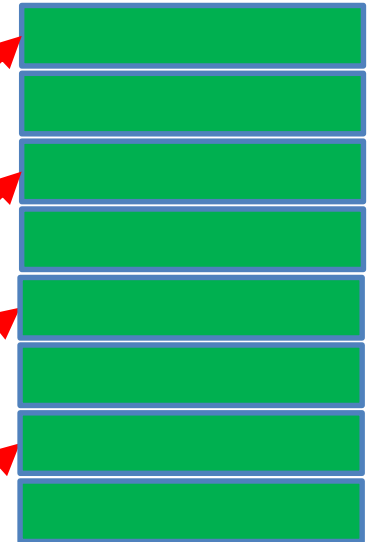
Process A

응용프로그램 주소값	가상메모리 주소
0x1111	0x1
0x2222	0x2
0x3333	0x3
0x4444	0x4
0x5555	0xa
0x6666	0xb

페이지 테이블

각 프로세스 마다 하나씩!

물리메모리를 논리 주소공간에 Map!!



RAM

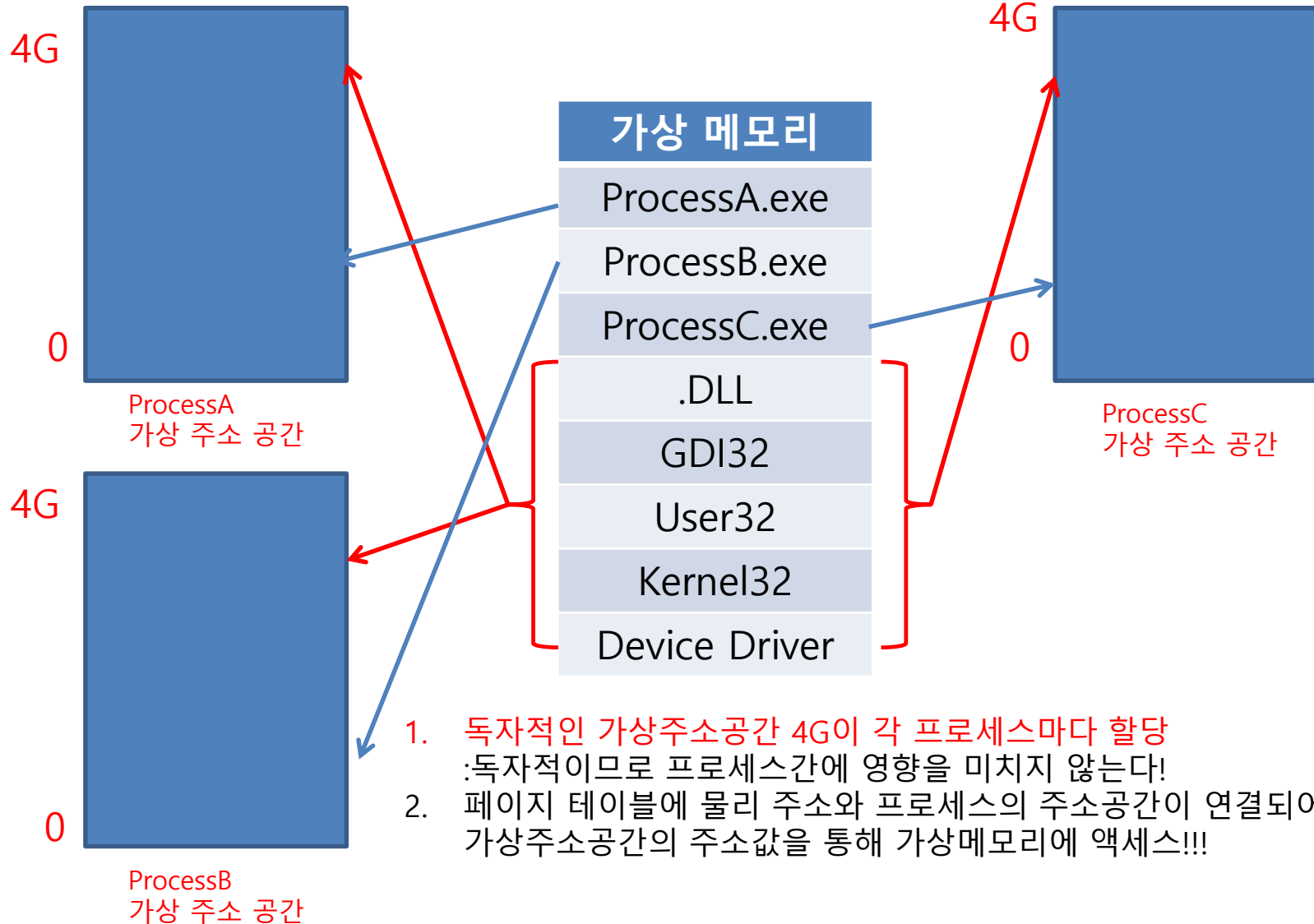


HARD DISK

우리가 VS에서 실제로 보는 주소값!!!

가상메모리(Virtual Memory)

3. 우리가 말하는 메모리의 정체



이것만 이해하면 C를 이해한 것

```
1  #include <stdio.h> //헤더파일
2
3  int AddTwoNum(int a, int b); //함수의 선언
4
5  int main(void) //os가 프로세스 실행시 호출
6  {
7      int num1 = 5; //지역변수1
8      int num2 = 10; //지역변수2
9
10     int num3 = AddTwoNum(num1, num2); //함수 호출
11
12     return 0; //누구에게 리턴?
13 }
14
15 int AddTwoNum(int a, int b) //매개변수 a, b
16 {
17     int c = a + b; //지역변수3
18
19     return c; //반환 시 과연 지역변수 c의 값을 반환할까?
20 }
```

함수

$y = f(x)$

f: 함수이름
x: 매개변수
y: 결과값(반환값)

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      unsigned char num = 256;
6      printf("%d \n", num);
7
8      return 0;
9  }
```

반환형 함수이름(매개변수)

```
int main(void)
```

```
{
```

```
    //code;    함수 몸체
```

```
    return 0;  반환값
```

```
}
```


함수의 선언과 정의

```
2  
3 int AddTwoNum(int a, int b); //함수의 선언
```

함수의 선언

인터페이스

- 어떤 반환형과 어떤 매개변수형을 가지는가?

Ex)

int ABC(int bbb, int ccc) 와 int DEF(int ddd, int eee)는 같은 인터페이스이다.

함수의 선언은 인터페이스를 지정하는 것

```
15 int AddTwoNum(int a, int b) //매개변수 a, b  
16 {  
17     int c = a + b; //지역변수  
18  
19     return c; //반환 시 과연 지역변수 c의 값을 반환할까?  
20 }
```

함수의 정의

함수의 정의

- 함수 몸체를 구현

매개변수와 반환값의 비밀

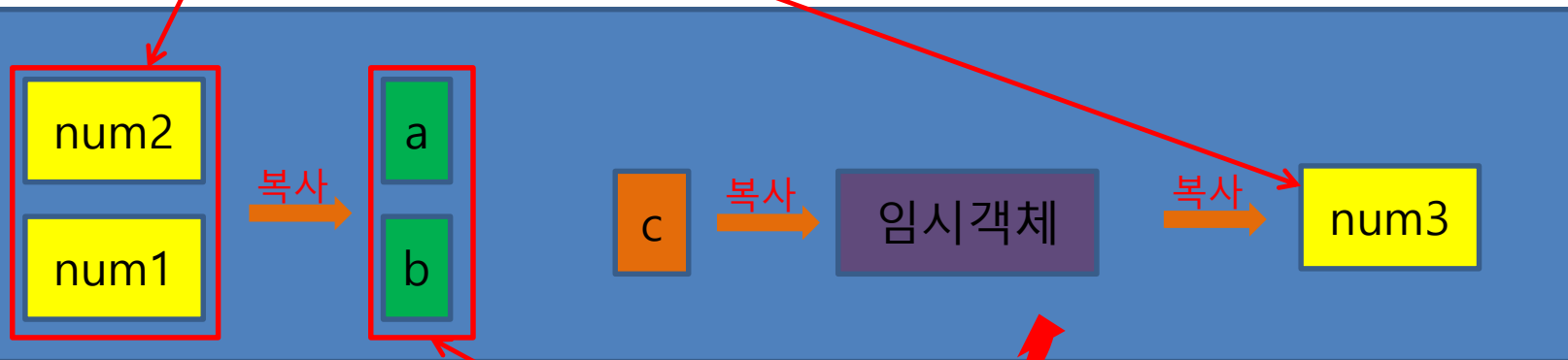
```

7 | int num1 = 5; //지역변수1
8 | int num2 = 10; //지역변수2
9 |
10 | int num3 = AddTwoNum(num1, num2); //함수 호출
11 |

```

함수 호출 시
함수에 전달되는 매개변수는
num1, num2가 아니다!!!!!!

STACK 영역



```

15 | int AddTwoNum(int a, int b) //매개변수 a, b
16 | {
17 |     int c = a + b; //지역변수3
18 |
19 |     return c; //반환 시 과연 지역변수 c의 값을 반환할까?
20 | }

```

스cope(Scope)

```

15 int AddTwoNum(int a, int b)//매개변수 a, b
16 {
17     int c = a + b;//지역변수
18
19     return c;//반환 시 과연 지역변수 c의 값을 반환할까?
20 }

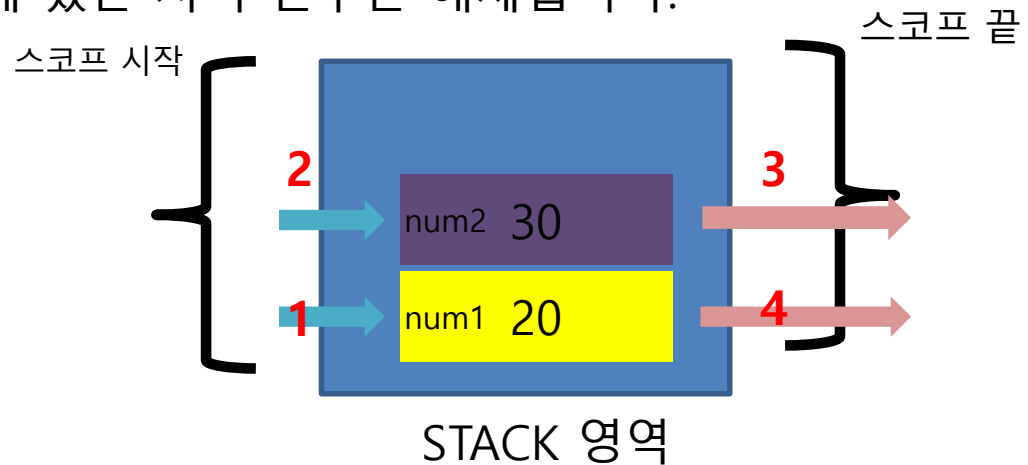
```

{ ... } 구문을 하나로 묶는 영역표시
이 영역을 지나면 STACK에 있는 지역 변수는 해제됩니다.

```

{
    int num1 = 20;
    int num2 = 30;
}

```



printf() 함수 1

- 입출력 함수 기본

```
int main(int argc, char * argv[])
{
    printf("argc = %d \n", argc);

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

    return 0;
}
```

1. 문자열을 화면에 출력
2. 문자열 안에 다양한 자료형을 넣을 수 있다.
 - 1) int 정수 : %d (d: decimal)
 - 2) char 문자 : %c (c: character)
 - 3) double, float : %f(기본적으로 double이지만 float로 출력해도 데이터 손실이 없으므로 함께 씁니다.)

잘 기억해두세요! scanf() 함수를 배울 때 유용합니다!

Scanf() 함수 1

- 입출력 함수 기본

```
int main(void)
{
    float fn;
    double dn;
    int num;

    scanf("%f %lf %d", &fn, &dn, &num);

    printf("%f %f %d", fn, dn, num);

    return 0;
}
```

& 연산자의 비밀은 포인터를 배우면
자연스럽게 풀리게 됩니다!

1. 문자열을 키보드에서 읽어들이.
2. 문자열 안에 다양한 자료형을 넣을 수 있다.
 - 1) int 정수 : %d (d: decimal)
 - 2) char 문자 : %c (c: character)
 - 3) double : %lf
 - 4) float : %f

서식문자가 printf()와 비슷하지만 다릅니다!!

Header 파일이란?

함수를 호출하려면 함수 선언이 필요합니다.
printf()와 scanf() 함수는 어디에 선언되어 있을까요?

```
1 #include <stdio.h>//헤더파일
```

헤더파일

- 변수 선언, 함수 선언 등을 모아놓은 파일
- C/C++은 선언과 정의를 따로 나누어 담는 경우가 대부분
- C++에서는 보통 헤더파일만 보고 프로그램의 전체적인 흐름을 판단해볼 수 있습니다.!


헤더파일의 종류

- `#include <stdio.h>`
: 표준 라이브러리
- `#include "Test.h"`
: 프로그래머가 만든 헤더파일

main() 함수

- int main(void)

프로세스 실행 시,
운영체제(OS)가 호출
하나만 존재 가능



```
5 int main(void) // OS가 프로세스 실행시 호출
6 {
7     int num1 = 5; // 지역변수1
8     int num2 = 10; // 지역변수2
9
10    int num3 = AddTwoNum(num1, num2); // 함수 호출
11
12    return 0; // 누구에게 리턴?
13 }
```

main() 함수의 인터페이스는 정해져 있다.

1. int main(void)
2. int main(int argc, char * argv[])

main()함수

- int main(int argc, char * argv[])

```
int main(int argc, char * argv[])
{
    printf("argc = %d \n", argc);

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

    return 0;
}
```

1. argc : 인자로 전달된 문자열의 수

포인터는 나중에!

1. argv : 전달된 문자열을 가리키는 포인터

기본 연산자

: 대입 연산자와 산술 연산자

연산자	연산자 기능	결합방향
=	오른쪽 값을 왼쪽 변수에 저장	←
+	두 피연산자를 더한다	→
-	왼쪽 연산자에서 오른쪽 연산자를 뺀다	→
*	두 피연산자를 곱한다	→
/	왼쪽 연산자를 오른쪽으로 나눈다	→
%	나누었을 때 나머지(ONLY 정수형 피연산자만) ex) 10.0f % 1.0f (컴파일 에러!)	→

기본 연산자
: 대입 연산자와 산술 연산자

```
int main(void)
{
    int n1 = 34;
    int n2 = 8;
    int result = n1 + n2;
    printf("result 값은 %d 입니다.\n", result);
    printf("나머지는 %d 입니다.\n", n1%n2);
    return 0;
}
```

연산결과
레지스터에 임시저장 후
result에 저장

먼저 연산이 일어난 후
printf 함수가 실행된다.
n1, n2 모두 정수이다.

연산자

: 우선 순위와 결합방향

$$7 + 3 - \boxed{2 \times 15} \quad \text{우선 순위}$$

$$\boxed{7 + 3} - 30$$



결합 방향

: 산술 연산자는 결합방향이 왼쪽에서 오른쪽으로 이동

$$10 - 30 = 20$$

만약 우선 순위가 같지만 결합 방향이 다른 두 연산자가 있다면??

즉, $7 + 3 - 30$ 에서

$+$, $-$ 가 우선 순위는 같되 결합방향이 달랐다면??

연산자는 우선 순위가 같으면 결합방향도 같다!!!

연산자

: 연산의 순서를 지정할 땐 ()

```
int num = 7 + (3 - 30)
```

구분자(separator)

우선 순위, 결합 방향에 상관없이 순서 지정

우선 순위를 외울 필요는 없고

가독성을 높이기 위해 필요한 경우 () 구분자를 많이 씁니다!!

다양한 연산자
: 복합 연산자

```
int a = 5;  
int b = 10;  
  
a += b;    a = a + b;  
a -= b;    a = a - b;  
a *= b;    a = a * b;  
a /= b;    a = a / b;
```

a += b;
-> a = a + b;

a -= b;
-> a = a - b;

a *= b;
-> a = a * b;

a /= b;
-> a = a / b;

다양한 연산자
: 증가 감소 연산자

전위 증가 연산자

```
int num1 = 6;  
int num2;  
  
num2 = ++num1;
```

- 1) num1이 1 증가
- 2) num2에 num1 값 대입

num2 == 7

후위 증가 연산자

```
int num1 = 6;  
int num2;  
  
num2 = num1++;
```

- 1) Num2에 num1 값 대입
- 2) num1이 1 증가

num2 == 6

다양한 연산자
: 증가 감소 연산자

```
int num1 = 6;  
int num2;  
  
num2 = (num1++) + (num1++);
```

Num2 값은??

```
int num = 6;  
  
num = num++;
```

Num 값은??

컴파일러마다 다르다!!!

다양한 연산자
: 관계 연산자

```
int num1 = 5;  
int num2 = 3;  
  
int result1 = (num1 == num2);  
int result2 = (num1 != num2);  
int result3 = (num1 > num2);  
int result4 = (num1 < num2);  
int result5 = (num1 >= num2);  
int result6 = (num1 <= num2);
```

== : 두 연산자가 같은가?

!= : 두 연산자가 같지 않은가?

다양한 연산자
: 논리 연산자

연산자	기능	결합방향
&&	둘 다 참이면 참 (논리 AND)	→
	둘 중 하나라도 참이면 참 (논리 OR)	→
!	참이면 거짓, 거짓이면 참 (논리 NOT)	←

비트 연산자와는 다른 연산자!!

다양한 연산자
: 논리 연산자

```
int num1 = 5;
int num2 = 3;

int result1 = (num1 == 5) && (num2 == 3);
int result2 = (num1 > 5) || (num2 < 4);
int result3 = !result1;
int result4 = !num1;

printf("result1 : %d\n", result1);
printf("result2 : %d\n", result2);
printf("result3 : %d\n", result3);
printf("result4 : %d\n", result4);
```

다양한 연산자
: 논리 연산자

```
int result1 = (num1 == 5) && (num2 == 3);
```

1 && 1

1

```
int result2 = (num1 > 5) || (num2 < 4);
```

0 || 1

1

```
int result3 = !result1;
```

!1

0

```
int result4 = !num1;
```

!5

0 : 거짓
정수 : 참

0

다양한 연산자
: 형 변환 연산자

```
char num1 = 5;  
int num2 = (int)num1;
```

1byte -> 4byte
범위가 넓은 쪽으로 변환

```
int num3 = 5;  
char num4 = (char)num3;
```

4byte -> 1byte
Data loss error!

```
float num5 = 3.14f;  
double num6 = (double)num5;
```

4byte -> 8byte
범위가 넓은 쪽으로 변환

```
double num7 = 3.14;  
float num8 = (float)num7;
```

8byte -> 4byte
Data loss error!

다양한 연산자
: 형 변환 연산자

```
int num9 = 11;  
float num10 = (float)num9;
```

4byte -> 4byte
정수 -> 실수
실수 -> 정수
변경도 가능

대입연산은 무조건 왼쪽이 기준

```
int num = 3.14;
```

대입하려는 값이 3.14로 double 형이지만

왼쪽 lvalue가 int 형이므로

형 변환되어 3이 저장

Lvalue : 대입연산의 왼쪽에 올 수 있는 연산자
대입이 가능한 연산자이므로 거의 변수일 확률이 높다

sizeof()

```
int main(void)
{
    printf("int : %d \n", sizeof(int));
    printf("char : %d \n", sizeof(char));
    printf("short : %d \n", sizeof(short));
    printf("long : %d \n", sizeof(long));
    printf("float : %d \n", sizeof(float));
    printf("double : %d \n", sizeof(double));

    return 0;
}
```

자료형 크기를 byte 단위로 알려준다.