KG 아이티뱅크

C 언어

C LANGUAGE

메모리 구조

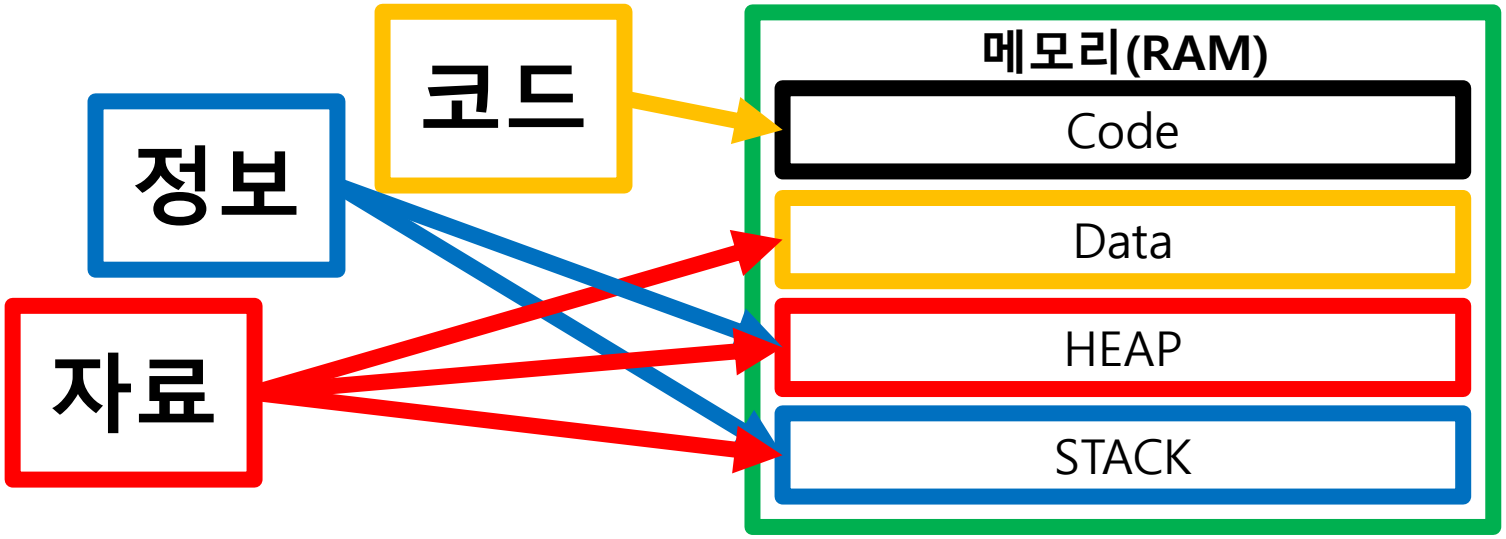
메모리 구조

- **기본적으로 컴퓨터는 모든 것을 저장하여 사용**
  - **자료** : 측정된 값들의 집합
  - **정보** : 이를 가공하여 만들어 내는 것
  - **코드** : 특정한 작업을 하도록 구성된 명령 목록
- **저장되는 곳은 주기억장치(RAM)의 임의의 공간**
  - 막무가내로 사용하면 공간이 꼬임
- **공간을 구분해 사용 : 메모리 분할(Memory Segmentation)**
  - 공간을 얼마만큼 필요한지 파악해 할당을 받아 쓰는 것
  - 공간은 여러 개가 될 수 있지만 **절대 공유하지 않음**

메모리 구조

- ❖ **저장해 사용할 수 있는 공간을 받았지만 여전히 구분해야 함**
  - ➢ **저장해서 사용하는 것**을 마구잡이로 배치할 수 없음
- ❖ **메모리 분할로 할당을 받은 공간을 다시 한번 나눔**
  - ➢ 코드(Code) 영역 : 수행하는 기능이 저장되는 공간
  - ➢ 데이터(Data) 영역 : 공통으로 사용할 수 있는 공간
  - ➢ 힙(HEAP) 영역 : 실시간으로 사용할 수 있는 공간
  - ➢ 스택(STACK) 영역 : 사용량을 미리 정해 놓은 공간

✓ **분할된 영역은 서로 간섭할 수 없으며 공유할 수 없음**



기본구조

기본구조

❖ 특정 프로그래밍 언어로 작성된 수행할 내용 : 코드

❖ 프로그래밍 언어로 수행할 내용을 작성하는 것 : 코딩

❖ 코딩을 효율적으로 하기 위해 정해져 있는 규칙 : 문법

① 사람의 언어와 유사한 특성을 가지고 있음

- 띄어쓰기, 엔터 등을 이용해 정리할 수 있음
- 문장의 끝을 알리기 위해 마침표가 존재함

② 다르기 때문에 사람의 언어로 설명이 필요

- 주석(Comment)기능으로 메모를 작성함

❖ 유사한 특성

```
printf("출력한다!");
int num;
num = 10;
double db=3.14;
```

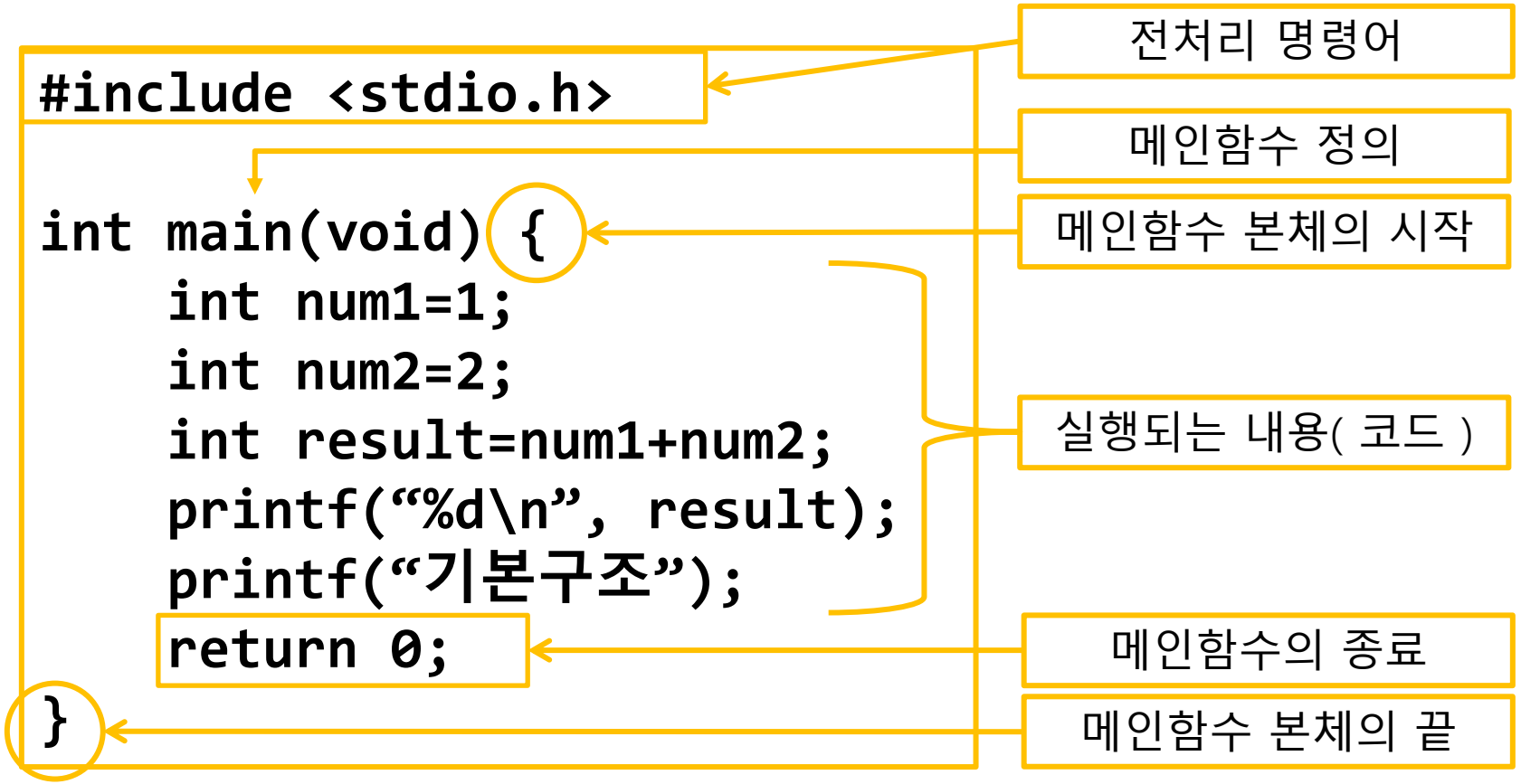
❖ 주석 기능 사용

```
// 한 줄 주석
/* 구간 주석
여러 줄 가능 */
```

기본구조

❖ C언어의 문법은 메모리를 사용하는 계획을 구성하는 것

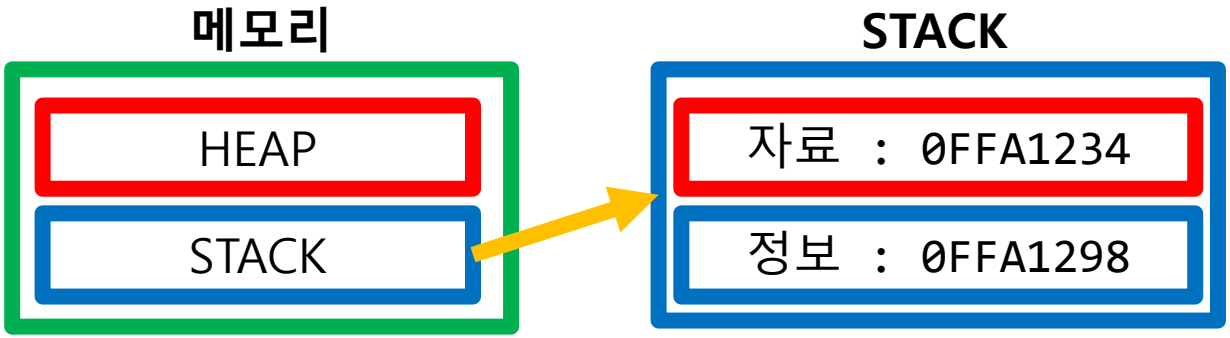
➢ 작성하는 내용을 적재적소에 저장하기 위한 규칙



변수

변수(Variable)

- **컴퓨터에 저장한 '것' 은 모두 주기억장치에 보관됨**
  - **어디에 있는지** 알아야 사용할 수 있음
  - **이게 무엇인지**를 알아야 사용할 수 있음
- **어디에 있는지에 대한 정보 : 메모리 주소**
  - 일반적으로 **2진수 32자리 ~ 64자리**로 구성
  - 너무 길이서 이를 **16진수 8자리 ~ 16자리로** 압축
- **이게 무엇인지에 대한 정보 : 자료의 종류**
  - 메모리를 어떤 구조로 얼마만큼 사용할 것인가를 의미

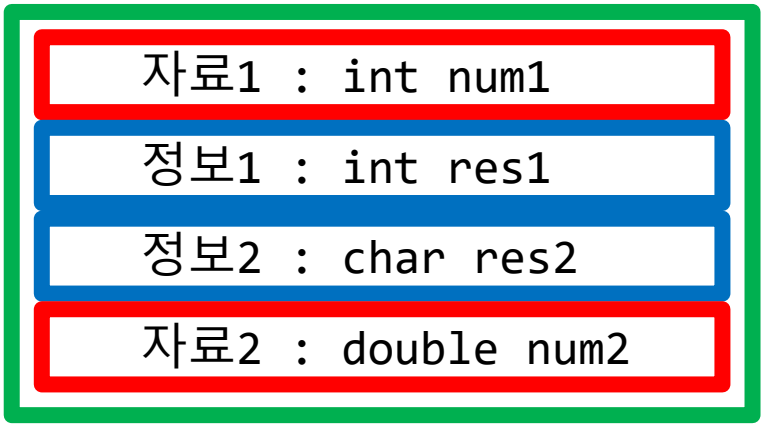


변수(Variable)

❖ 문제 : 메모하거나 기억하기에는 너무 어려움

1) 저장된 위치를 파악해야 함
2) 읽을 크기를 지정해야 함
3) 하나라도 틀리면 망함
4) **결론 : 비효율적인 작업**

❖ 해결 : 이름과 속성을 부여하여 쉽게 관리를 시도



1) 저장된 위치 몰라도 됨
2) 위치 대신 이름을 사용
3) 이름에는 속성 정보가 존재
4) **결론 : 효율적인 작업**

변수(Variable)

❖ 상수 : 이름을 붙이지 않고 사용하는 것

❖ 속성은 문법에 따라 자동으로 붙음

  ❖ 필요한 곳에 바로 작성하여 사용

❖ 준비된 공간은 위치를 몰라 사용 불가

  ➢ 값의 **교체가 불가능**

❖ 상수가 될 수 없는 것
1) 문법적으로 잘못된 것
2) 문법적으로 잘못된 것

❖ 변수 : 이름을 붙여 사용하는 것

❖ 이름을 통해 속성을 붙여 관리하는 것

  ➢ 중복된 값을 동일한 곳에서 불러 사용

❖ 준비된 공간은 계속 사용할 수 있음

  ➢ 값의 **교체가 가능**

❖ 저장할 수 있는 값
➢ 문법에 맞게 만든 상수
➢ 다른 변수에 저장된 값

변수(Variable)

❖ **자료형 : 변수에 설정된 값의 속성**

① **변수에 한번 부여된 자료형은 바꿀 수 없음**

② 자료형을 통해 **값의 종류를 식별함**

③ 자료형을 통해 **필요한 만큼의 공간을 확보함**

자료형	크기(byte)	범위
char	1	-128 ~ 127
short	2	-32768 ~ 32768
int	4	-2147483648 ~ 2147483647
long	4	-2147483648 ~ 2147483647
float	4	3.4E +/- 38 (**유효 6자리**)
double	8	1.7E +/- 308 (**유효 15자리**)

변수(Variable)

❖ **변수의 생성** : 변수를 **선언**한다

➢ 저장공간을 요청해 속성과 이름을 붙임

```
❖ 변수의 선언 : 자료형 변수명;
int num;    // 선언
```

❖ **변수에 최초로 값을 저장** : 변수를 **초기화**한다

➢ **최초로 저장할 때 한정**으로 맞춰서 정리하는 과정

```
❖ 변수의 초기화 : 변수 = 값
num = 10;   // 초기화
```

❖ **이후의 값을 저장** : 변수에 값을 **저장**한다

➢ **기존값을 파괴하고 새로운 값으로 갱신**하는 과정

```
❖ 변수의 초기화 : 변수 = 값
num = 15;   // 저장
```

변수(Variable)

❖ 변수의 생성 규칙

1) 영문자(대소문자 구별), 숫자, 언더바(_)만 사용
   - ➢ 한글 등의 다른 언어는 사용하지 말 것
2) 변수명 내에 공백(White Space) 사용 불가
   - ➢ 띄어쓰기를 하면 안됨
   - ➢ 띄어쓰기가 필요하면 언더바(_) 를 사용
3) 변수명의 첫 문자는 영문자 혹은 언더바(_)로 시작
   - ➢ 숫자로 시작하면 숫자 상수로 인식하게 됨
4) 중복된 이름이 존재하면 선언할 수 없음
5) C언어의 예약어는 변수명으로 사용 불가
   - ➢ 너무 많아서 외우는 것은 의미 없음

❖ C언어의 예약어

auto	double	int
struct	break	else
long	switch	case
enum	register	typedef
char	extern	return
union	const	float
short	unsigned	continue
for	signed	void
default	goto	sizeof
volatile	do	if
static	while	

변수(Variable)

- ❖ **상수 : 변수에 저장하거나 바로 사용하는 값**
- ❖ **문법만 맞다면 값의 종류가 자동으로 식별됨**

1) **문자 상수 : 언어**를 표현하기 위한 값
   - ① **단일문자 상수** : 'A', '1', 'C', 'F', '*'
   - ② **문자열 상수** : "가", "ABC", "^^", "1"
2) **숫자 상수 : 수치**를 표현하기 위한 값
   - ① **정수형 상수** : 123, 10, 011(8), 0x18(16)
   - ② **실수형 상수** : 3.14, 2.72

❖ 올바른 상수의 작성

'A'
123
3.14
"문자열 상수!"
"ABC***DEF"

❖ 부적절한 상수의 작성

'A"
1 23
3 .14
'문자열 상수!'
"AAA***DEF'

변수(Variable)

- ❖ **변수의 선언, 초기화 등은 모두 프로그래머의 재량에 달린 것**
  - ➢ 변수가 많이 필요하다면 많이 선언하면 됨
  - ➢ 변수의 초기화가 필요가 없다면 방치해도 되지만 권장하진 않음
- ❖ **작성하는 코드에 필요한 만큼의 변수를 선언하고 사용해야 함**
  - ➢ **경험이 관여하는 문제**이기 때문에 경험이 쌓일수록 해결됨
- ❖ **변수를 선언할 때 가장 주의해야 하는 사항 : 변수의 이름**
  - ➢ 너무 길거나 너무 짧으면 작성 및 이해하기 어려움
  - ➢ 지나친 약자를 사용하면 이해하기 어려움
- ✓ **적절하게 의미를 전달할 수 있는 단어 단위로 권장**

num1 : 첫번째 숫자 변수
float_num : 실수 변수
name : 이름이 저장될 변수
result : 결과가 저장될 변수

what_is_this
n_o_n
a, b ,c, A
➢ 의미 불명