

1인1프로그래밍언어 마스터제

# 집중프로그래밍언어(코딩클리닉) 교육

전남대학교 SW중심대학사업단

# 1인1프로그래밍언어 마스터제 소개

- 목적

- ✓실전적 SW개발역량 강화를 위하여 SW전공(소프트웨어공학과, 컴퓨터 정보통신공학과, AI융합학부)대상으로 코딩클리닉 교육

- 운영

- ✓학기중 프로그래밍언어(C/C++, Java, Python) 교과목 운영


- 교육일정

- 일시: 2024. 09. 26(목) 16:00 ~ 18:00
  - 장소: AI융합대학 210호실

- 교육자료

- URL: <https://github.com/yoojaemyeong/cprogram>

# 깃허브 자료실 (<https://github.com/yoojaemyeong/cprogram>)


 **cprogram** Public




main

1 Branch

0 Tags

Go to file

 **yoojaemyeong** Add files via upload

 README.md	Update README.md
 검색.ipynb	Add files via upload
 연결리스트.ipynb	Add files via upload

README

## 1인1프로그래밍언어 마스터제

**C 프로그래밍(구조체와 포인터)**  
실전적 SW개발 역량 강화를 위하여 SW전공 대상으로 코딩클리닉 교육

**교육일시**  
2024. 9. 26(목) 16:00 ~ 18:00

**교육장소**  
AI융합대학 210호

# C 프로그램 실습 환경 만들기

1. VirtualBox 설치(지원되는 버전: 4.0 ~ 7.0)

- <https://www.virtualbox.org>

2. Vagrant 설치

- <https://www.vagrantup.com>

3. 제공한 Vagrant 스크립트 실행 순서

① 로컬PC 작업폴더 생성 및 이동: `mkdir c:\cprogram; cd c:\cprogram`

② Vagrant 초기환경 만들기: `vagrant init`

③ 제공한 Vagrantfile.txt의 내용 복사하여 Vagrantfile에 붙여넣기

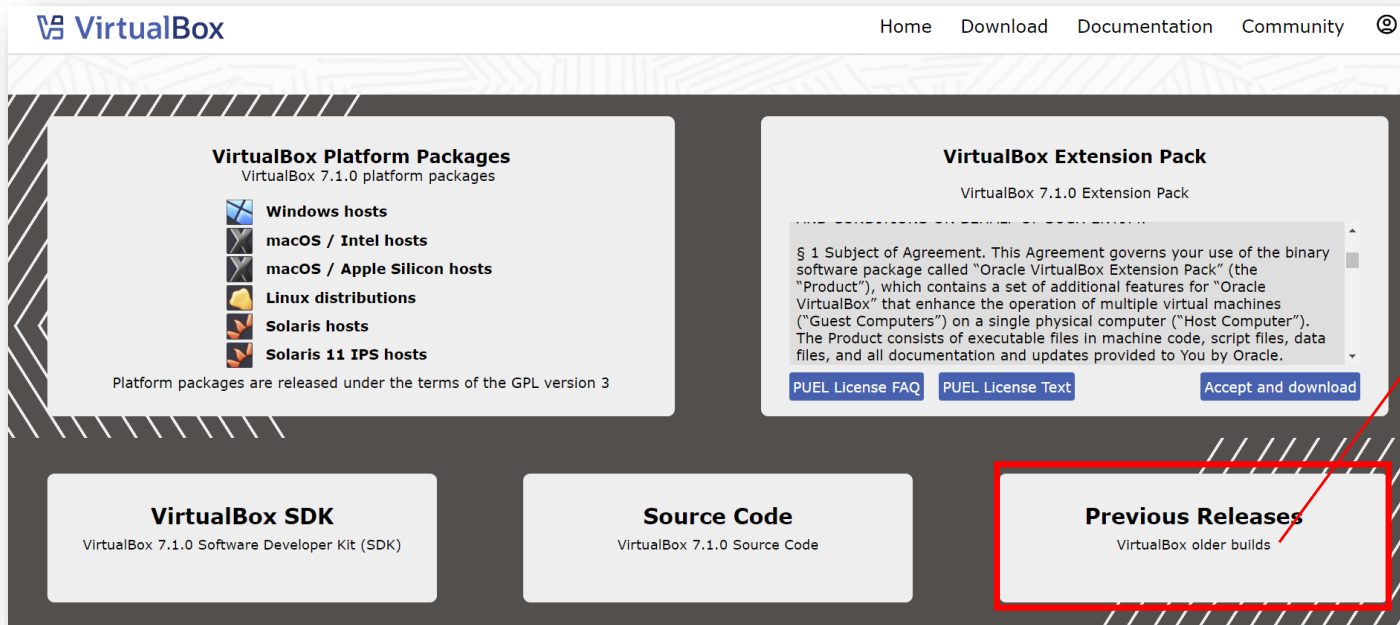
④ 프로그램 설치 시작: `vagrant up`

❖ 설치과정 중에 네트워크 접근 허용 알림창이 뜨는데 허용할 것

⑤ 명령창에서 접근: `vagrant ssh`

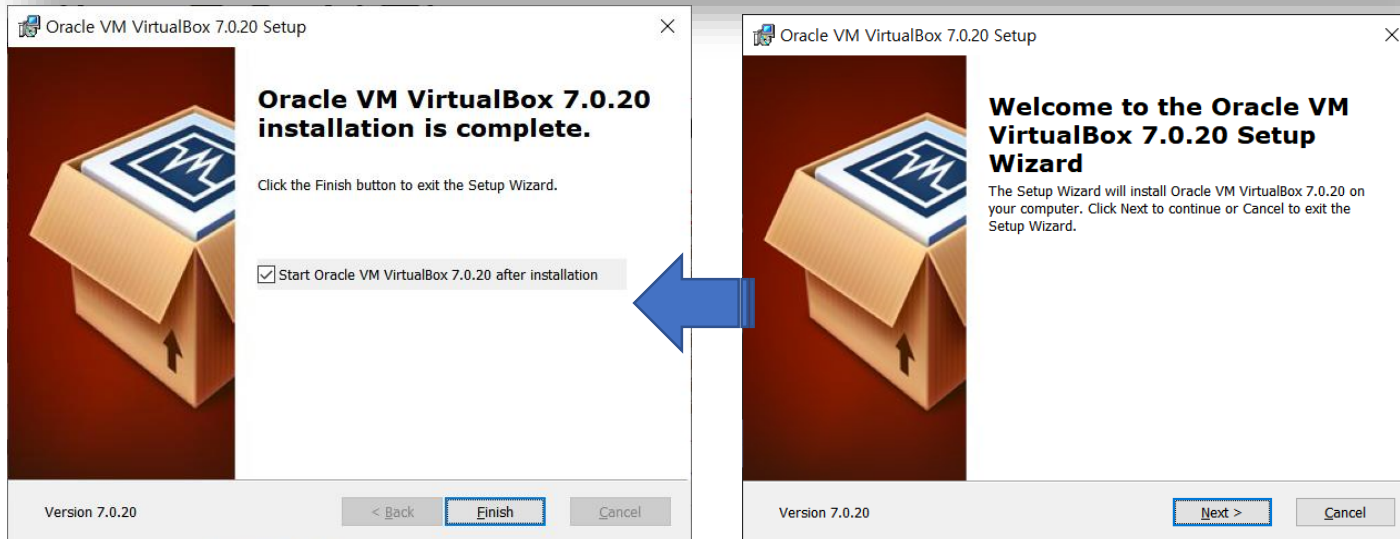
⑥ 로컬PC의 작업폴더에 있는 TOKEN\_URL.txt 파일 열어 URL 경로로  
브라우저에서 열기

# Virtualbox7.0 설치



## Download VirtualBox (Old Builds)

- **VirtualBox 7.0 (active maintenance)**
- VirtualBox 6.1 (**no longer supported**, support ended 2024/01)
- VirtualBox 6.0 (**no longer supported**, support ended 2020/07)
- VirtualBox 5.2 (**no longer supported**, support ended 2020/07)
- VirtualBox 5.1 (**no longer supported**, support ended 2018/04)
- VirtualBox 5.0 (**no longer supported**, support ended 2017/05)
- VirtualBox 4.3 (**no longer supported**, support ended 2015/12)
- VirtualBox 4.2 (**no longer supported**, support ended 2015/12)
- VirtualBox 4.1 (**no longer supported**, support ended 2015/12)
- VirtualBox 4.0 (**no longer supported**, support ended 2015/12)

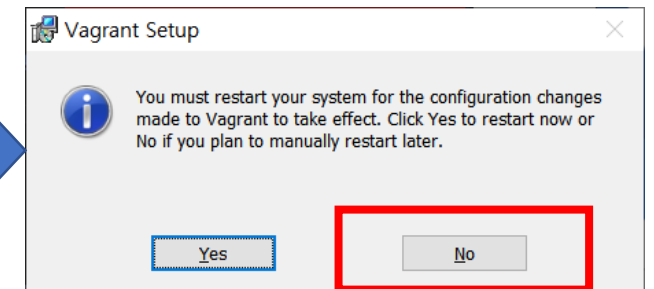
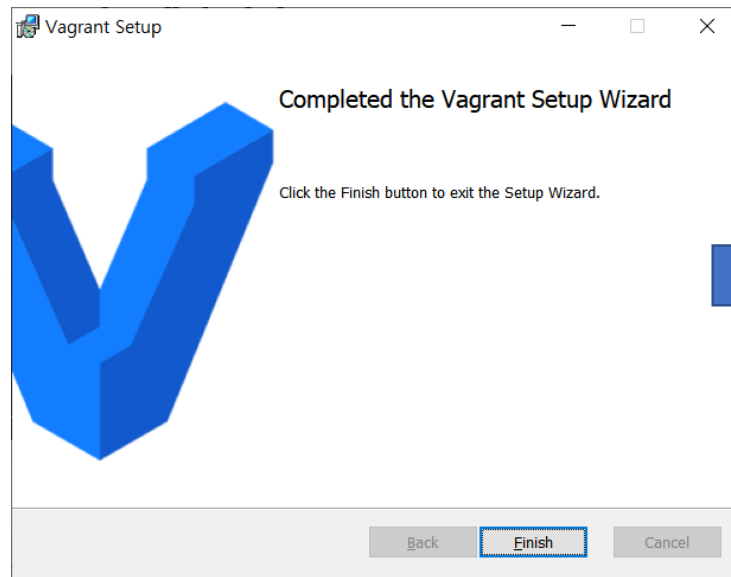
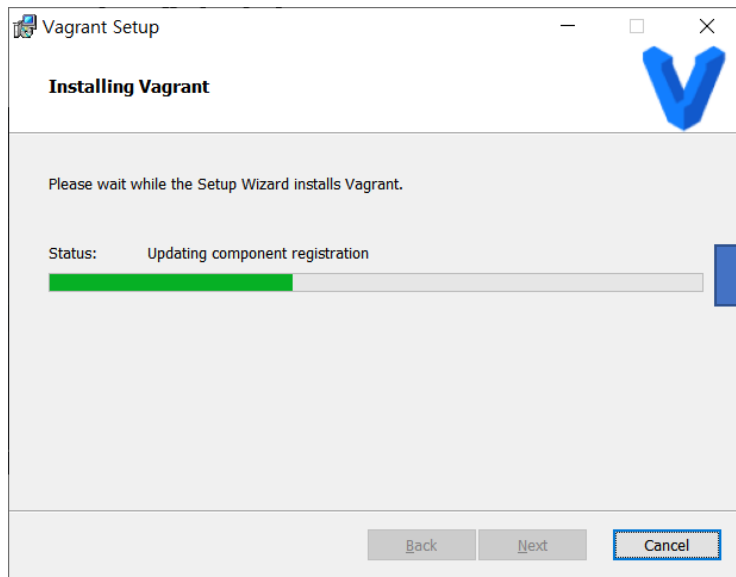
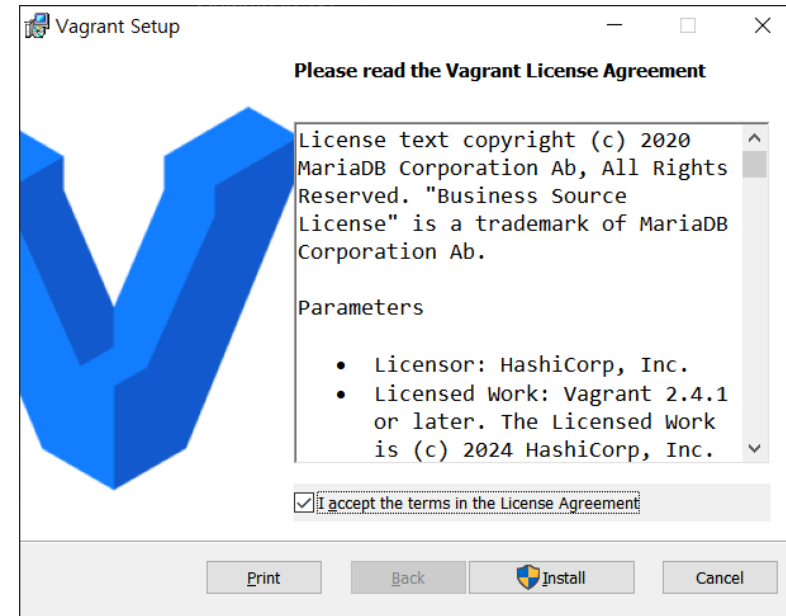
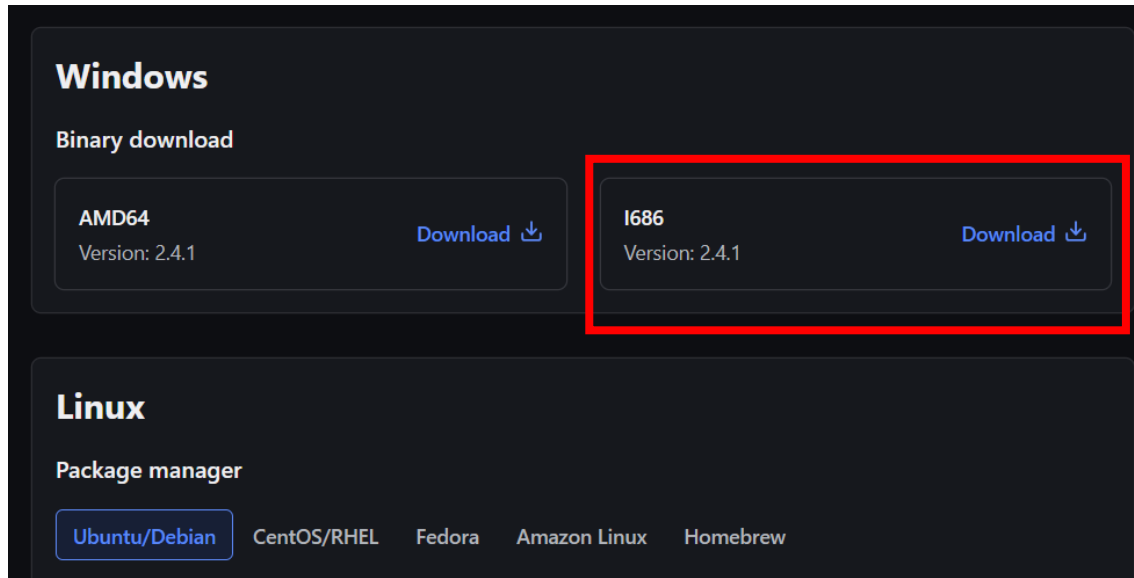


## VirtualBox

The Extension Packs in this section are released under the Oracle VM VirtualBox License. All other binaries are released under the terms and conditions of the respective licenses.

- 7.0 SDK (7.0.20)
- **VirtualBox 7.0.20 (released July 16 2024)**
  - Windows hosts
  - macOS / Intel hosts
  - Solaris hosts
  - Solaris 11 IPS hosts

# Vagrant 설치



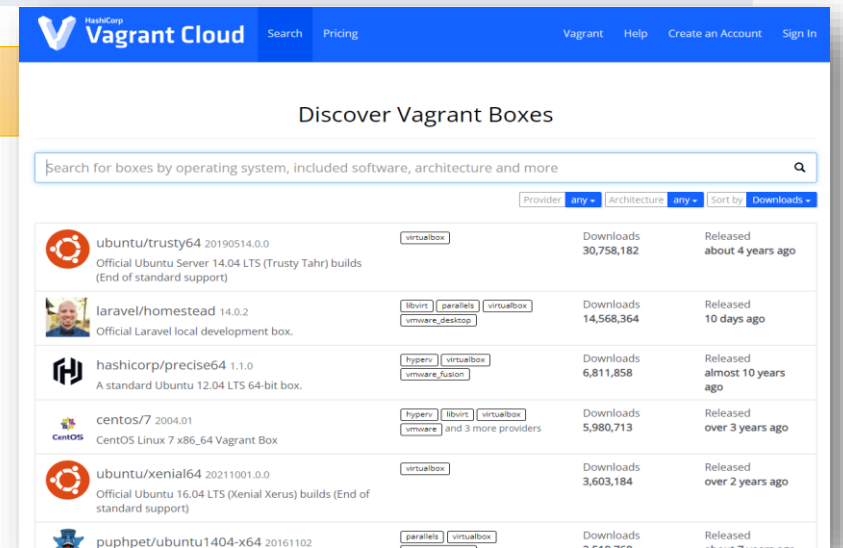
No 선택 및 완료

# Vagrant 명령어 및 vagrant 이미지 저장소

명령어	설명
vagrant init	프로비저닝을 위한 기초 파일 생성
vagrant up	Vagrantfile을 읽어들이며 프로비저닝 시작
vagrant halt	Vagrant 가상 머신 종료
vagrant destroy	Vagrant 관리대상 가상 머신 삭제
vagrant ssh	Vagrant 관리 가상 머신에 ssh 접속
vagrant provision	Vagrant 관리하는 가상 머신에 변경된 설정을 적용

Vagrant 클라우드

<https://app.vagrantup.com/boxes/search>



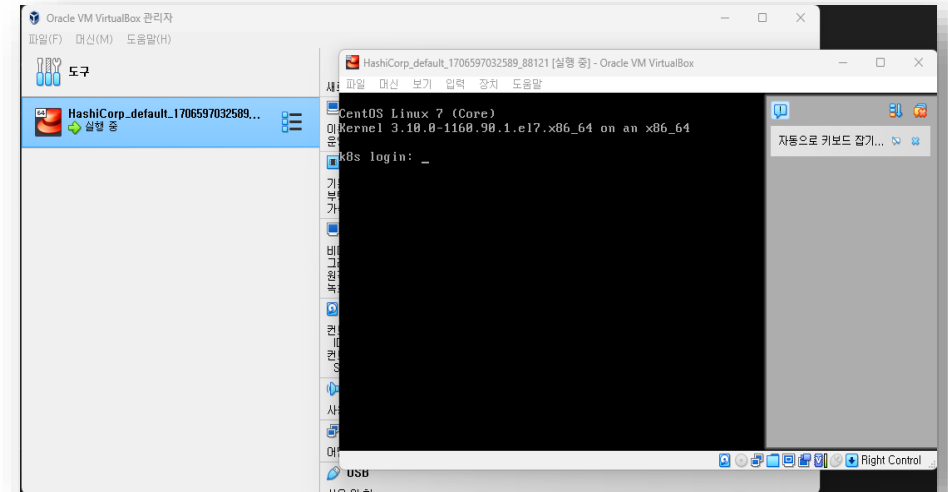
# Vagrant 활용 VM 생성

- **vagrant init** : 초기환경 생성( Vagrantfile )

vi Vagrantfile

```
# -*- mode: ruby -*-  
# vi: set ft=ruby :  
  
Vagrant.configure("2") do | config |  
  config.vm.box = "sysnet4admin/CentOS-k8s"  
end
```

- **vagrant up** : virtualbox에 vm 생성
- **vagrant ssh** : CentOS에 ssh 연결
  - 패스워드 : vagrant
- **vagrant destroy -f** : VM 삭제





# Vagrant 구성파일

## VM 만들기

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
  #vagrant허브에서 이미지 가져오기 및 네트워크 환경
  config.vm.box = "bento/ubuntu-22.04"
  config.vm.network "private_network", ip: "192.168.5.10"
  config.vm.network "forwarded_port", guest: 8888, host: 60888, auto_correct:true,id:"http"
  config.vm.host_name = "cprogram"
  # 호스트와 게스트간 폴더 공유
  config.vm.synced_folder "C:\\w\\cprogram", "/home/vagrant/SharedProjects"
  #[게스트]VM 하드웨어 사양
  config.vm.provider "virtualbox" do |vb|
    vb.name = "cprogram"
    vb.gui = true
    vb.cpus = 2
    vb.memory = 2048
    vb.customize ["modifyvm",:id,"--groups","/UbuntuGroup"]
  end
end
```

## VM내에 docker 및 jupyterlab 생성 스크립트

```
#[게스트]VM 초기 설치환경
config.vm.provision "shell", inline: <<-SHELL
  mkdir /home/vagrant/SharedProjects
  apt-get -y update
  apt-get -y upgrade
  sudo apt-get install -y apt-transport-https ca-certificates gnupg-agent software-properties-
common
  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
  sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable" -y
  sudo apt-get update -y
  sudo apt-get install -y docker-ce docker-ce-cli containerd.io
  sudo usermod -aG docker vagrant
  sudo chmod 666 /var/run/docker.sock
  echo -n '#!/usr/bin/env bash
  apt-get -y update
  sudo --user=jovyan pip install jupyterlab
  sudo --user=jovyan conda install -c conda-forge jupyterlab jupyterlab-git
  sudo --user=jovyan pip install jupyter-cpp-kernel
  ln -s /workspace /home/jovyan/workspace
  chmod -R 777 /workspace
  chown -R jovyan:users /workspace' > /home/vagrant/SharedProjects/addperm.sh
  echo -n '#!/usr/bin/env bash
  TOKEN=$(docker exec -i jupyter jupyter lab list | tail -1 | cut -d: -f3)
  TOKEN2=$(echo $TOKEN | cut -d= -f2)
  echo
"http://localhost:60888/?token=$TOKEN2" > /home/vagrant/SharedProjects/TOKEN_URL.txt
' > /home/vagrant/SharedProjects/jupyter-info.sh
  chmod 755 /home/vagrant/SharedProjects/jupyter-info.sh
  docker run -itd -e GRANT_SUDO=yes -e TZ=Asia/Seoul --user root --name jupyter -v
/etc/localtime:/etc/localtime:ro -v /home/vagrant/SharedProjects:/workspace -p 8888:8888 --
restart=always jupyter/datascience-notebook
  docker exec -i jupyter sh /workspace/addperm.sh
SHELL
end
```

## 프로그램 실행: vagrant init/up

```
F:\2024-2-c-programming>vagrant init
A 'Vagrantfile' has been placed in this directory. You are now
ready to 'vagrant up' your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
'vagrantup.com' for more information on using Vagrant.
```

```
F:\2024-2-c-programming>vagrant up
```

## 명령프롬프트에서 접근: vagrant ssh

```
F:\2024-2-c-programming>vagrant ssh
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-83-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Sun Jul 28 08:42:08 PM UTC 2024

System load:  0.02          Processes:           158
Usage of /:   39.6% of 30.34GB Users logged in:          0
Memory usage: 27%          IPv4 address for eth0: 10.0.2.15
Swap usage:   0%

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento
vagrant@npm-java:~$
```

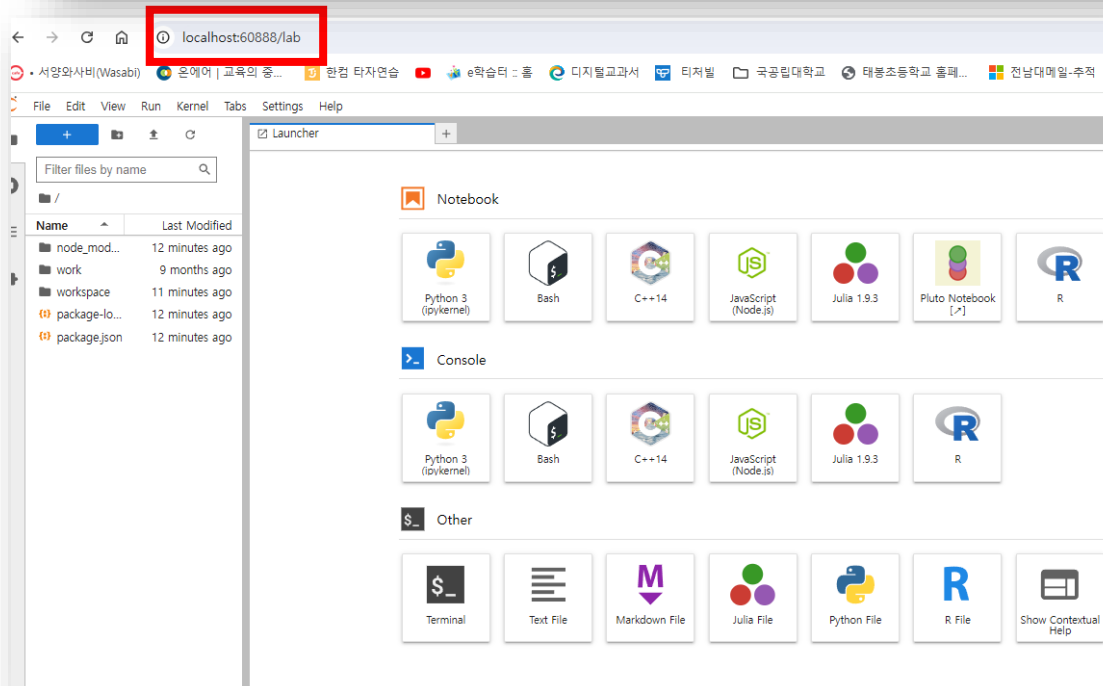
기본 패스워드 : **vagrant**

```
vagrant@cprogram:~$ cd SharedProjects/
vagrant@cprogram:~/SharedProjects$ sh jupyter-info.sh
```

로컬PC의  
작업폴더

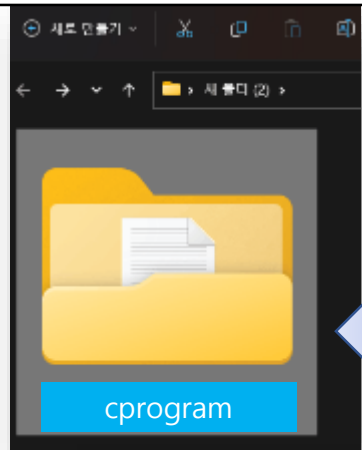
.vagrant  
addperm  
jupyter-info  
TOKEN\_URL  
Vagrantfile

TOKEN\_URL - Window 메모장  
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)  
<http://localhost:60888/?token=190ee2be67e8328e2666d66f9a>



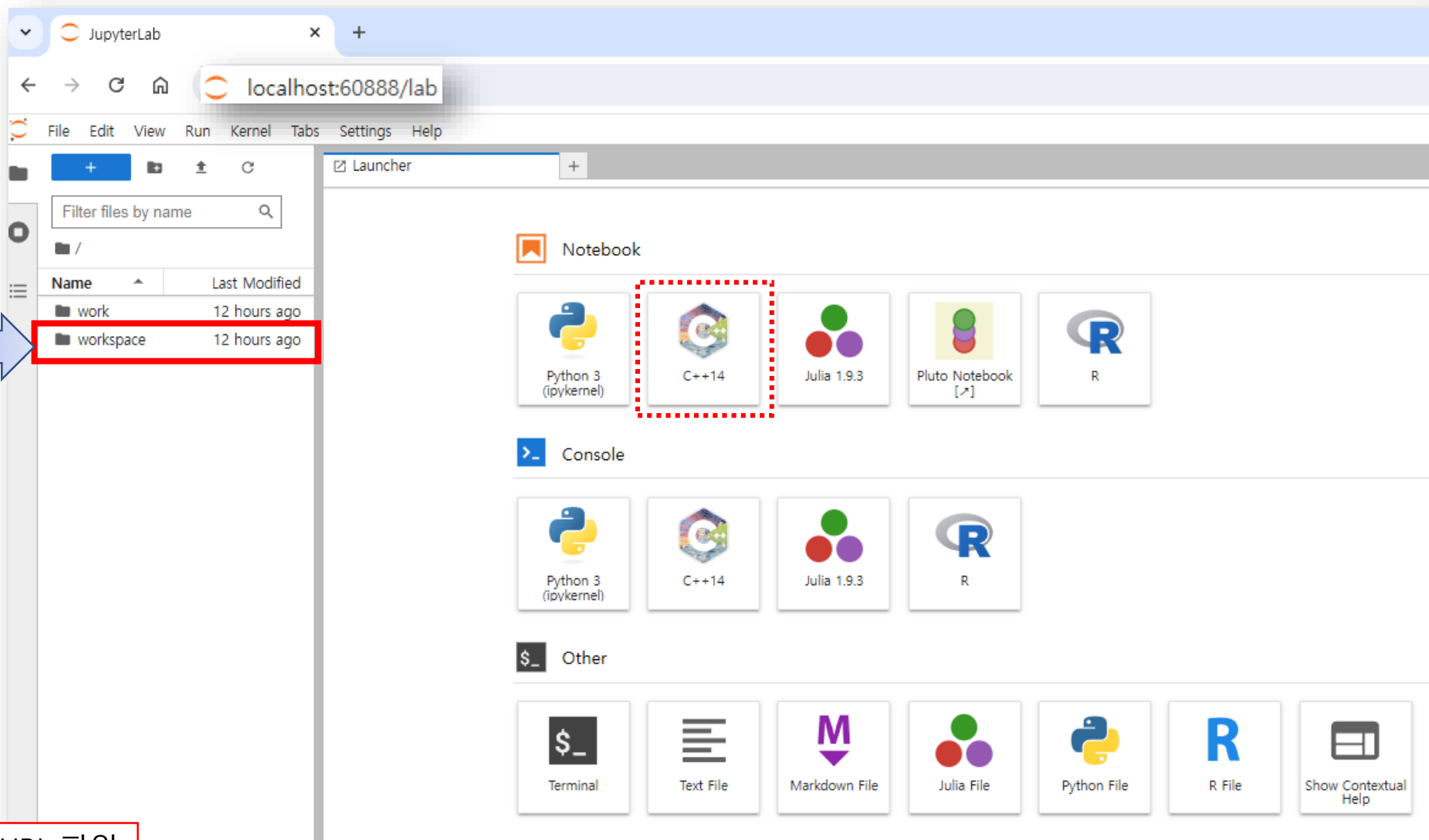
## 웹브라우저에서 주피터노트북 화면

로컬PC의 작업폴더



.ipynb\_checkpoints  
.vagrant  
Project1  
addperm.sh  
jupyter-info.sh  
test-c.ipynb  
Vagrantfile  
TOKEN\_URL.txt

주피터노트북 로그인 정보 URL 파일



# 실행화면

localhost:60889/lab/tree/workspace/struct\_pointer.ipynb

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/ workspace /

Name	Last Modified
Jupyterlab-...	now
Project1	9 months ago
addperm.sh	5 days ago
jupyter-inf...	5 days ago
struct_poin...	29 seconds ago
test-c.ipynb	11 days ago
TOKEN_UR...	5 days ago
Vagrantfile	5 days ago

```
[3]: struct Student {
      char name[20];
      int age;
      float grade;
    };
    struct Student std1 = {"yoo",20,4.0};

    printf("std.name = %s", std1.name);

std.name = yoo

[9]: #include <stdlib.h>
#include <stdio.h>

struct Student {
    char name[20];
    int age;
    float grade;
};
int main(){
    struct Student *ptr = (struct Student*)malloc(sizeof(struct Student));
    strcpy(ptr->name, "boy");
    ptr->age = 22;
    ptr->grade = 3.8;
    printf("Name: %s, Age: %d, Grade: %.2f\n",ptr->name, ptr->age, ptr->grade);
    free(ptr);
}

Name: boy, Age: 22, Grade: 3.80

[ ]:
```

# 구조체란?

- 기본데이터 타입(**stdio.h**에서 정의)

- 문자형: char
- 정수형: int, short, long, long long
- 실수형: float, double
- bool형으로 true, false 저장 (**stdbool.h**에서 정의 by C99 표준)

```
#include <stdio.h>
#include <stdbool.h>
int main(){
    bool isTrue = true;
    bool isFalse = false;
    if(isTrue) printf("이것은 true\n");
    if(isFalse==0) printf("이것은 false\n");
}
```

- 사용자정의 데이터 타입

- 사용자정의 변수 선언: typedef
- 구조체: struct
- 공용체: union
- 열거체: enum

```
struct Student {
    char name[50];
    int age;
    float grade;
};
```

```
typedef struct [TagName]
{
    Type1 Name1;
    Type2 Name2;
    ...
    TypeN NameN;
} TypeName;
```

- 구조체는 사용자정의 데이터타입으로 여러 데이터타입을 그룹으로 묶어 활용

# 구조체 변수

```
#include <stdio.h>
typedef struct STag {
    int m;
} SType;
int main(){
    // struct 변수 선언: Tag이름
    struct STag s1;
    s1.m = 1;

    // struct 변수 선언: typedef 이용
    SType s2;
    s2.m = 2;

    printf("s1.m = %d\ns2.m = %d\n",s1.m, s2.m);
}
```

```
#include <stdio.h>
typedef struct {
    int m1;
    int m2;
    double m3;
} SType;
int main(){
    SType s = { 1, 2, 3.3 }; // 구조체 초기화
    printf("m1 = %d, m2 = %d, m3 = %.2f",s.m1, s.m2, s.m3);
}
```

# 구조체 크기

- 크기 측정 : sizeof

```
#include <stdio.h>
typedef struct {
    char c[6];
    int i;
    double d;
} SType;

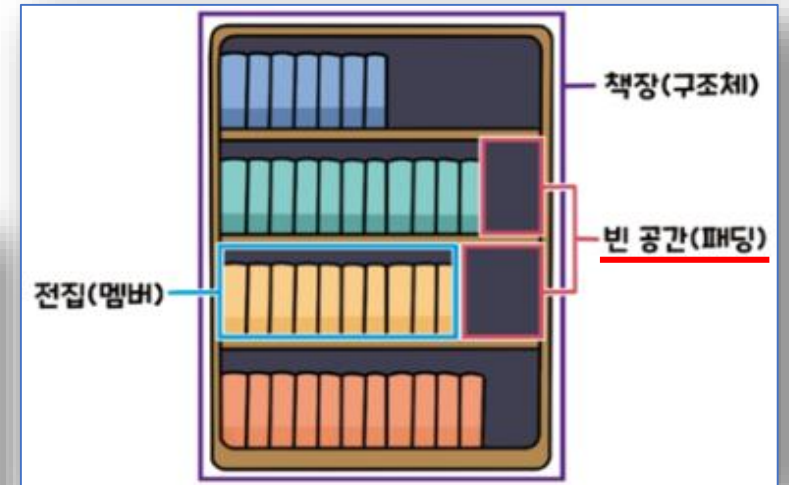
int main(){
    int size1 = sizeof(char) + sizeof(int) + sizeof(double); // 기본 타입들의 크기
    int size2 = sizeof(SType); // 구조체 크기

    printf("size1: %d, size2: %d",size1,size2);
}
```

size1: 13, size2: 24

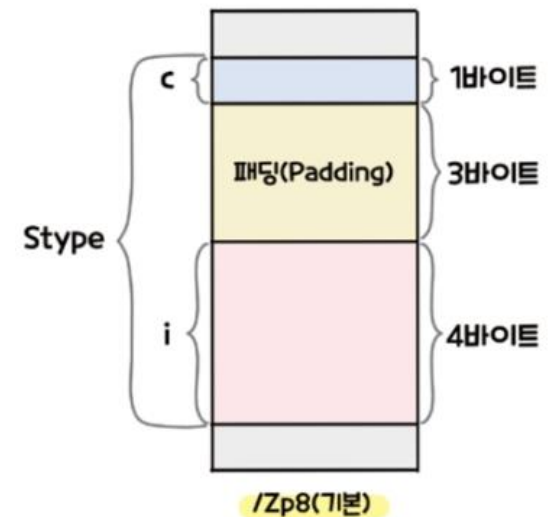
## 구조체 멤버와 패딩의 개념

\* 메모리 영역을 해석할 때 읽거나 쓰지 않는 부분



## 구조체 메모리 구조 : /Zp8

\* 컴파일러가 기본으로 선택한 최적 절충 지점



# 구조체 크기

## 구조체 패딩 부가 설명

- 구조체 선언

```
struct box {
    char c;
    long long ll;
}
```

- 32bit OS는 한 번에 4byte씩 메모리를 읽기 때문에 3번 접근 필요
- 64bit OS는 한 번에 8byte씩 메모리를 읽기 때문에 2번 접근 필요



- **32bit** : II에 접근하기 위해 메모리에 3번 접근
- **64bit** : II에 접근하기 위해 메모리에 2번 접근

## 이 구조체 형태에 패딩 적용



- 패딩 비트만큼 메모리 낭비
- 그러나 CPU 연산 횟수 감소로 인한 연산 속도 증가

```
#include <stdio.h>
int main(){
    struct box {
        char c;
        long long ll;
    };
    printf("box의 크기: %ld", sizeof(box)); //x64 타입
}
```

box의 크기: 16



# 포인터란?

- 메모리 주소를 저장하는 변수
  - 데이터가 저장되어 있는 위치

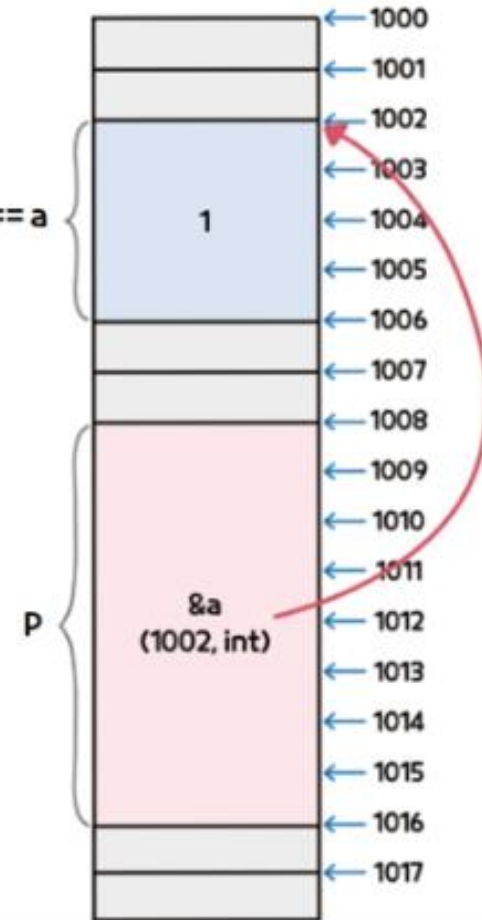
```
#include <stdio.h>
int main(){
    int a;
    int* p = &a;
    *p = 1;
    printf("a: %d(%p)\n*p: %d\n",a,p,*p);
}
```

a: 1(0x7ffc740d6cbc)

\*p: 1

간접(\*) 연산자 \*p == a

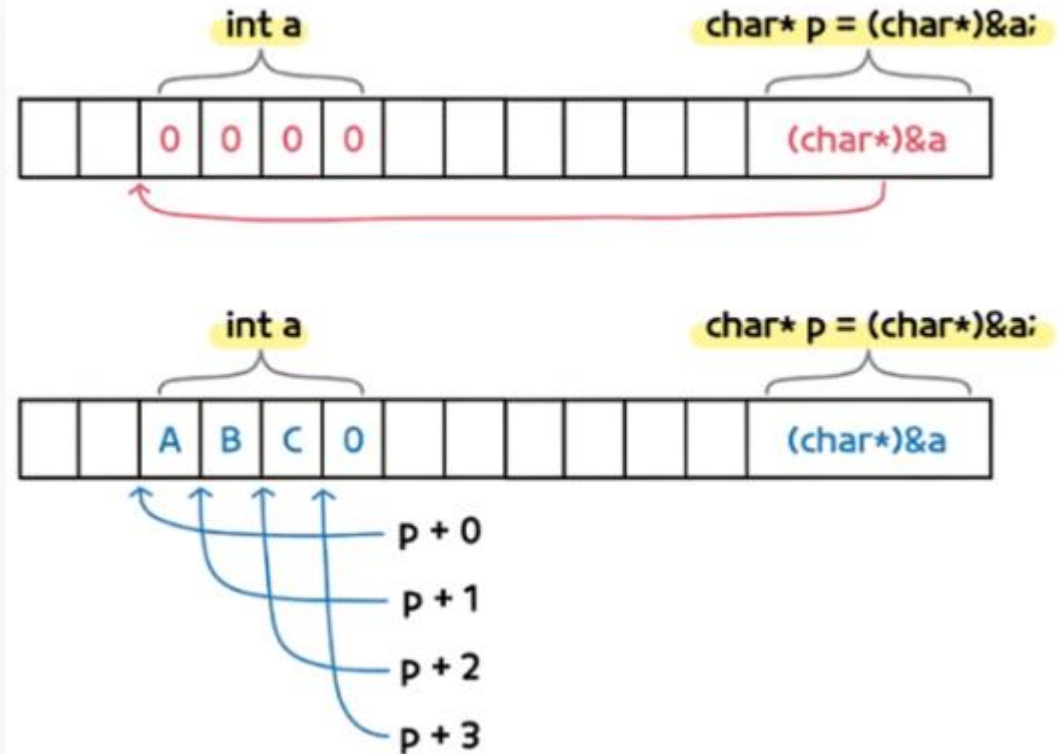
참조(&) 연산자 P



```
#include <stdio.h>
int main(){
    int a = 1;
    int *p = &a;
    printf("a = %d, sizeof(p) = %d", a, sizeof(p));    //포인터의 크기
}
```

# 대상 타입과 객체 타입이 다른 포인터

```
#include <stdio.h>
int main(){
    int a = 0;
    char* p = (char*)&a;
    *(p+0) = 'A';
    *(p+1) = 'B';
    *(p+2) = 'C';
    *(p+3) = '\\0';
    printf("%s == %s", (char*)&a, p);
    printf("\\n%p\\n", &a);
    for(int i = 0; i < 4; i++)
        printf("*p = %c\\n", *(p+i));
    *(p+0) = 100;
    *(p+1) = 0;
    *(p+2) = 0;
    *(p+3) = 0;
    for(int i = 0; i < 4; i++)
        printf("*p = %c\\n", *(p+i));
    printf("%d\\n", a);
}
```



# 구조체와 포인터

```
#include <stdio.h>
typedef struct {
    int m;
    char* str1;
    char* str2;
} SType;

int main(){
    SType s;
    SType *pS = &s;
    pS->m = 2024;
    pS->str1 = "Happy ";
    (*pS).str2 = "New year !!!";

    printf("s.m: %d, s.str1: %s, s.str2: %s",s.m,s.str1,pS->str2);
}
```

간접 멤버 연산자(->)

직접 멤버(.) 연산자

`pS->str2 = (*pS).str2`

s.m: 2024, s.str1: Happy , s.str2: New year !!!

# 구조체와 포인터를 활용한다.

```
#include <stdio.h>
struct Student {
    char name[20];
    int age;
    float grade;
};
int main(){
    struct Student std1 = {"Hong", 25, 9.9};
    printf("Name: %s, Age: %d, Grade: %.2f\n", std1.name, std1.age, std1.grade);
}
```

직접 멤버 연산자 .

Name: Hong, Age: 25, Grade: 9.90

```
#include <stdlib.h>
#include <stdio.h>
struct Student {
    char name[20];
    int age;
    float grade;
};
int main(){
    struct Student *ptr = (struct Student*)malloc(sizeof(struct Student));
    strcpy(ptr->name, "boy");
    ptr->age = 22;
    ptr->grade = 3.8;
    printf("Name: %s, Age: %d, Grade: %.2f\n", ptr->name, ptr->age, ptr->grade);
    free(ptr);
}
```

간접 멤버 연산자 ->

Name: boy, Age: 22, Grade: 3.80

# 메모리 할당과 동적 객체 생성

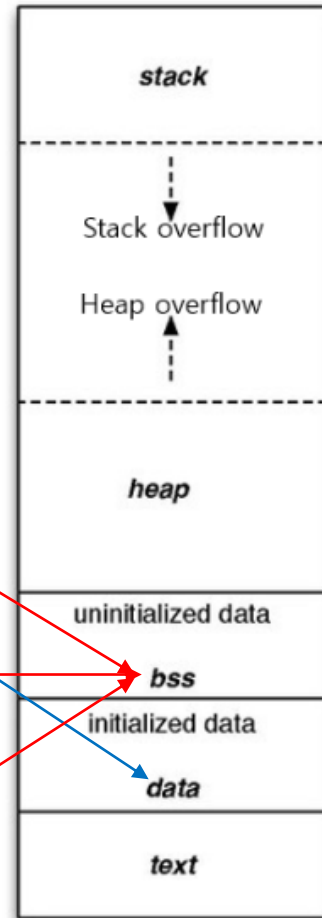
- 객체 가시 범위 및 생명 주기

	지역 객체	전역 객체	정적 객체 <code>static</code>	
			정적 지역 객체	정적 전역 객체
가시 범위	함수 블록	프로그램 전체	함수 블록	소스 파일
생명 주기	함수 호출~반환	프로그램 시작~종료	프로그램 시작~종료	프로그램 시작~종료

# 메모리 할당과 동적 객체 생성

## • 메모리 구조

```
int iBss;           // 전역변수, 초기화되지않음
int gData = 0;      // 전역변수, 초기화됨
static int siBss;   // 정적전역변수, 초기화되지않음
int main(){
    static int loBss; //정적지역변수, 초기화되지않음
}
```



지역변수, 매개변수, 복귀 번지 등 프로그램이  
자동으로 사용하는 임시 메모리 영역  
- LIFO(Last In First Out) 정책

동적 세그먼트

프로그래머가 동적으로 사용하는 영역  
- malloc, free 또는 new, delete에 할당, 반환

전역변수(global), 정적변수(static), 배열(array),  
구조체(structure) 등 저장  
- 런타임 이후 초기화 : BSS(Block Started by Symbol)  
- 런타임 이전 초기화 : data 영역

정적 세그먼트

코드 영역 - 작성한 코드, 프로세스가 종료될  
때까지 유지

# 메모리 할당과 동적 객체 생성

- 동적 객체 생성

## calloc 함수

헤더: #include <stdlib.h>

형식: void \*calloc(size\_t nmemb, size\_t size);

설명: size 크기의 자료가 nmemb개만큼 들어갈 메모리를 할당, 초기값이 0

반환: 성공하면 할당된 영역의 포인터 반환, 실패하면 NULL

참고로, NULL은  
stdio.h 에 매크로로 정의  
#define NULL((void \*)0)

참고로, size\_t은  
typedef unsigned \_\_int64 size\_t  
unsigned \_\_int64로  
부호없는 8바이트 정수 타입

## malloc 함수

헤더: #include <stdlib.h>

형식: void \*malloc(size\_t size);

설명: size 크기의 메모리를 할당, 초기값은 정의되지 않음

반환: 성공하면 할당된 영역의 포인터 반환, 실패하면 NULL

## free 함수

헤더: #include <stdlib.h>

형식: void free(void \*ptr);

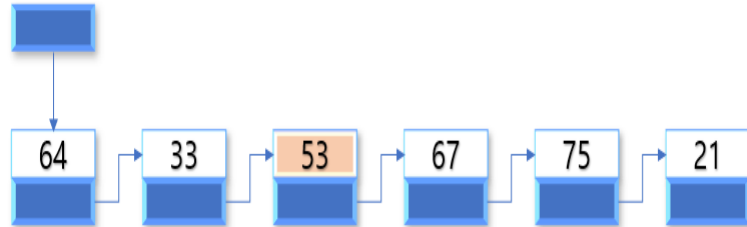
설명: ptr이 가리키는 메모리 해제

반환: 없음

검색알고리즘에서 포인터 활용



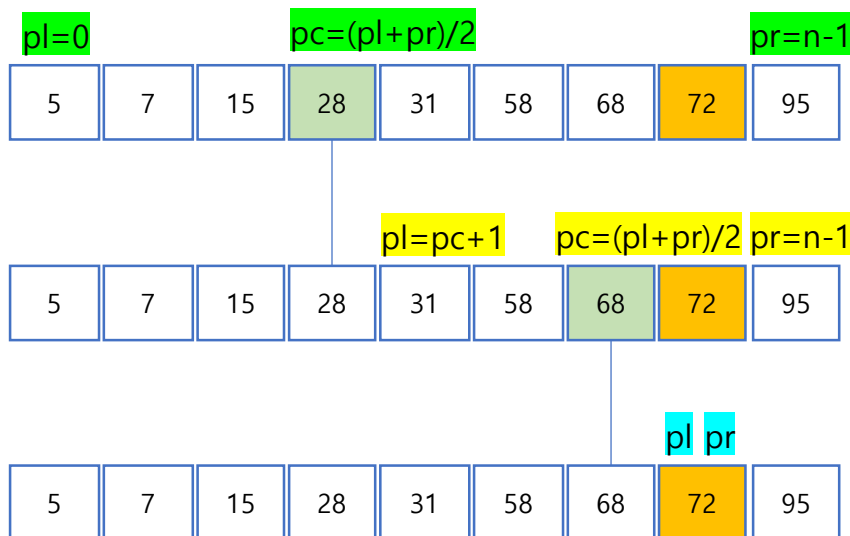
# 선형검색



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int search(const int a[], int n, int key){
4     int i = 0;
5     while(1){
6         if(i == n) return -1;
7         if(a[i] == key) return i;
8         i++;
9     }
10 }
11 int main(){
12     int nx, ky; // nx: 목록갯수, ky: 검색값
13     printf("선형검색\n■ 목록 갯수:");
14     scanf("%d", &nx);
15     int *x = (int *)calloc(nx, sizeof(int));
16     printf("\n 1 < [입력값 범위] < 100 \n");
17     for(int i = 0; i < nx; i++){
18         printf("x[%d] = ", i);
19         scanf("%d", x+i);
20     }
21     printf("■ 총 %d 개 검색 목록 생성 완료\n", nx);
22     printf("■ [선형검색] 검색값: ");
23     scanf("%d", &ky);
24     int idx = search(x, nx, ky);
25     if(idx == -1) printf("\n검색 실패\n");
26     else printf("\n■ 검색값(%d)은 x[%d] = %d\n", ky, idx, x[idx]);
27     free(x);
28 }
```

# 이진검색

이진검색은 요소가 오름차순  
또는 내림차순으로 정렬된  
배열에서 검색하는 알고리즘



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int bin_search(const int a[], int n, int key){
4     int pl = 0; // position left
5     int pr = n-1; // position right
6     do{
7         int pc = (pl+pr)/2;
8         if(a[pc] == key) return pc;
9         else if(a[pc] < key) pl = pc + 1;
10        else pr = pc - 1;
11    }while(pl <= pr);
12    return -1; //검색실패
13 }
14 int main(){
15     int nx, ky; // nx: 목록갯수, ky: 검색값
16     printf("이진검색\n■ 목록 갯수:");
17     scanf("%d", &nx);
18     int *x = (int *)calloc(nx, sizeof(int));
19     printf("\n[조건] 1<값<100 & 오름차순 입력\n");
20     printf("x[%d] = ", 0);
21     scanf("%d", x);
22     for(int i = 1; i < nx; i++){
23         do{
24             printf("x[%d] = ", i);
25             scanf("%d", x+i);
26             }while(x[i] < x[i-1]);
27     }
28     printf("■ 총 %d 개 검색 목록 생성 완료\n", nx);
29     printf("■ [이진검색] 검색값: ");
30     scanf("%d", &ky);
31     int idx = bin_search(x, nx, ky);
32     if(idx == -1) printf("\n검색 실패\n");
33     else printf("\n■ 검색값(%d)은 x[%d] = %d\n", ky, idx, x[idx]);
34     free(x);
35 }
```

# 함수 포인터

- 함수를 가리키는 포인터

예)

`double func(int);` //함수원형으로 매개변수 int형 인수로 받아 double형 값 반환

> 위 함수를 가리키는 포인터 fp 선언

`double(*fp)(int);` 또는 `double (*fp)(int);` // O

`double *fp(int);` // X

# 함수 포인트

- 함수를 가리키는 포인트

```
1 //함수를 가리키는 포인트
2 #include <stdio.h>
3 int sum(int x1, int x2){
4     return x1+x2;
5 }
6 int mul(int x1, int x2){
7     return x1*x2;
8 }
9 void func(int(*calc)(int,int)){
10     for(int i=1;i<10;i++){
11         for(int j=1;j<10;j++)
12             printf("%3d", (*calc)(i,j));
13         printf("\n");
14     }
15 }
16
17 int main(){
18     printf("덧셈표\n");
19     func(sum);
20     printf("곱셈표\n");
21     func(mul);
22 }
```

# 함수 포인트

## ❖ bsearch 함수 사용

헤더: #include <stdlib.h>

원형:

void \*bsearch(const void \*key, void \*base, size\_t nmemb, size\_t size, int(\*compar)(const void \*, const void \*));

설명:

자료(base)중

요소크기가 size인 nmemb 개수 중에서

compar 비교함수로 key를 검색

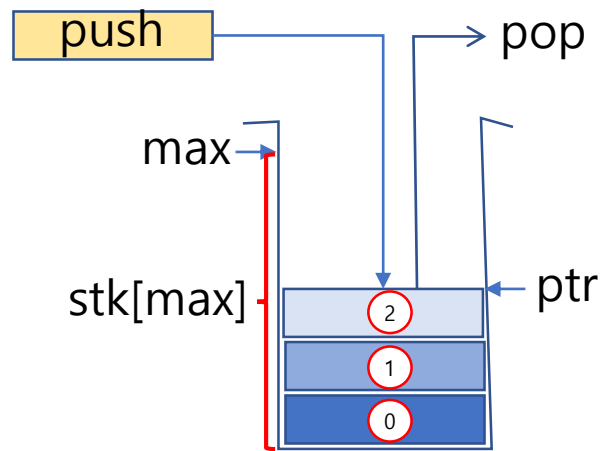
key와 일치하는 값을 찾으면 요소의 포인트 반환

없으면 NULL

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h> // string
4
5 typedef struct _Person{
6     char name[20]; //이름
7     int height;    //키
8     int weight;    //몸무게
9 } Person;
10 //비교함수
11 int npcmp(const Person *x, const Person *y){
12     return strcmp(x->name, y->name); // name이 같으면 0
13 }
14 int main(){
15     Person x[] = {
16         {"hong", 180, 77 },
17         {"kim", 183, 79},
18         {"yoo", 181, 75},
19         {"choi", 173, 60},
20         {"cha", 187, 77},
21         {"mike", 195, 85},
22     };
23     int nx = sizeof(x)/sizeof(Person);
24     int retry;
25     printf("■ 함수 포인트 활용\n");
26     do {
27         Person temp;
28         printf("찾는 이름: ");
29         scanf("%s", temp.name);
30         Person *p = (Person *)bsearch(&temp, x, nx, sizeof(Person), (int (*)(const void *, const void *)) npcmp);
31         if(p==NULL) printf("\n검색실패");
32         else printf("x[%d]: %s %d %d\n", (int)(p - x), p->name, p->height, p->weight);
33
34         printf("다시할까요?(1 or 0)");
35         scanf("%d",&retry);
36     }while(retry==1);
37 }
```

# 스택 Stack(1)

- LIFO, Last In First Out

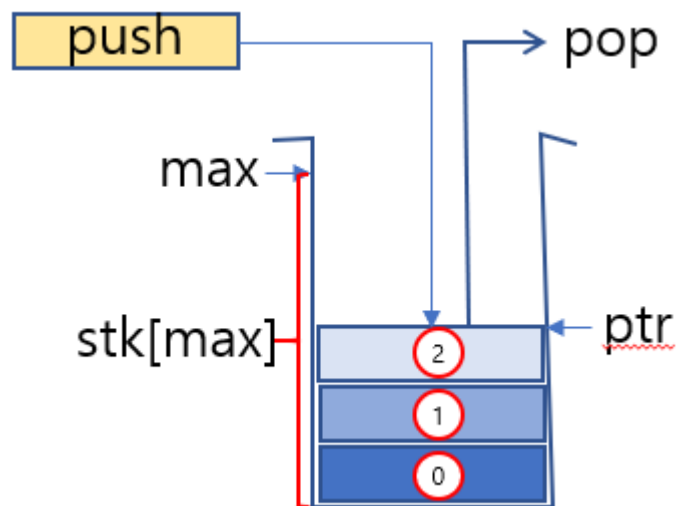


```
32 /* 스택 종료 */
33 void Terminate(IntStack *s){
34     if(s->stk != NULL) free(s->stk);
35     s->max = s->ptr = 0;
36 }
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 typedef struct {
4     int max;
5     int ptr;
6     int *stk;
7 } IntStack;
8 int Initialize(IntStack *s, int max){
9     s->ptr = 0;
10    if((s->stk = (int *)calloc(max, sizeof(int))) == NULL){
11        s->max = 0;
12        return -1;
13    }
14    s->max = max;
15    return 0;
16 }
17 int Push(IntStack *s, int x){
18     if(s->ptr >= s->max) return -1;
19     s->stk[s->ptr++] = x;
20     return 0;
21 }
22 int Pop(IntStack *s, int *x){
23     if(s->ptr <= 0) return -1;
24     *x = s->stk[--s->ptr];
25     return 0;
26 }
27 void Print(const IntStack *s){
28     int i;
29     for(int i = 0 ; i < s->ptr ; i++) printf("%d ",s->stk[i]);
30     printf("\n");
31 }
```

# 스택 Stack(2)

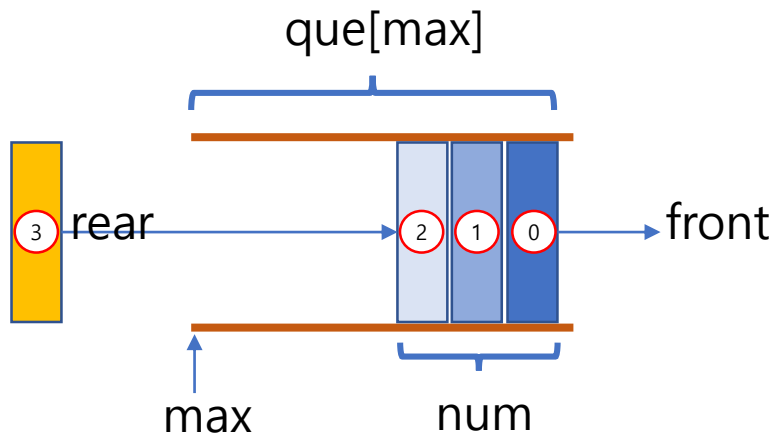
- LIFO, Last In First Out



```
37 int main(){
38     IntStack s;
39     if(Initialize(&s, 64) == -1){
40         printf("스택 생성 실패");
41         return 1;
42     }
43     printf("스택 시작\n");
44     while(1){
45         int menu,x; //데이터 x
46         printf("■ 스택프로그램\n (1) Push\n (2)Pop\n (3)출력\n (0)종료 \n선택하시오: ");
47         scanf("%d",&menu);
48         if(menu == 0) break;
49         switch(menu) {
50             case 1:
51                 printf("■ Push 정수 데이터: ");
52                 scanf("%d", &x);
53                 if(Push(&s,x) == -1) printf("Error: 푸시 실패");
54                 break;
55             case 2:
56                 if(Pop(&s, &x) == -1) printf("Error: 팝 실패");
57                 else printf("■ Pop 데이터: %d\n",x);
58                 break;
59             case 3:
60                 Print(&s);
61                 break;
62         }
63     }
64     Terminate(&s);
65 }
```

# 큐 Queue(1)

- Queue, First In First Out



```

43 /* 큐 종료 */
44 void Terminate(IntQueue *q){
45     if(q->que != NULL) free(q->que);
46     q->max = q->num = q->front = q->rear = 0;
47 }

```

```

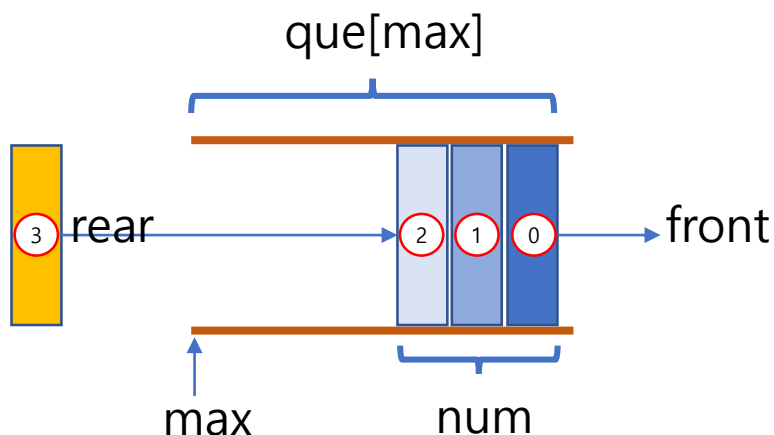
1 #include <stdio.h>
2 #include <stdlib.h>
3 typedef struct {
4     int max;
5     int num;
6     int *que;
7     int front;
8     int rear;
9 } IntQueue;
10 int Initialize(IntQueue *q, int max){
11     q->num = q->front = q->rear = 0;
12     if((q->que = (int *)calloc(max, sizeof(int))) == NULL){
13         q->max = 0;
14         return -1;
15     }
16     q->max = max;
17     return 0;
18 }
19 int Enqueue(IntQueue *q, int x){
20     if(q->num >= q->max) return -1;
21     else{
22         q->num++;
23         q->que[q->rear++] = x;
24         if(q->rear == q->max) q->rear = 0;
25         return 0;
26     }
27 }
28 int Dequeue(IntQueue *q, int *x){
29     if(q->num <= 0) return -1;
30     else{
31         q->num--;
32         *x = q->que[q->front++];
33         if(q->front == q->max) q->front = 0;
34         return 0;
35     }
36 }
37 void Print(const IntQueue *q){
38     int i;
39     printf("=====\n");
40     for(int i = 0 ; i < q->num ; i++) printf("%d ", q->que[(i+q->front) % q->max]);
41     printf("\n=====\n");
42 }

```



# 큐 Queue(2)

- Queue, First In First Out

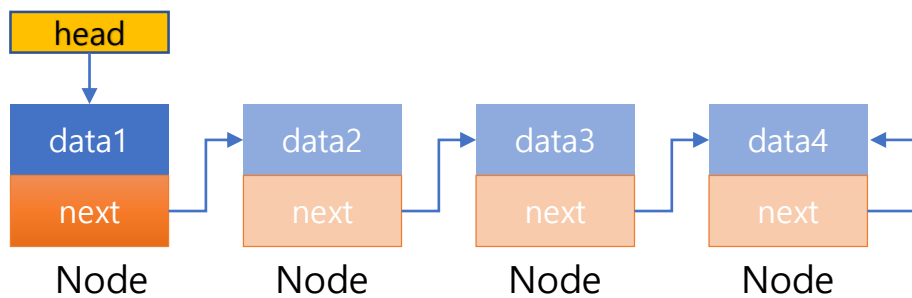


```
48 int main(){
49     IntQueue que;
50     if(Initialize(&que, 64) == -1){
51         printf("큐 생성 실패");
52         return 1;
53     }
54     printf("스택 시작\n");
55     while(1){
56         int menu,x; //데이터 x
57         printf("■ 큐프로그램\n (1) 인큐\n (2) 디큐\n (3)출력\n (0)종료 \n선택하시오: ");
58         scanf("%d",&menu);
59         if(menu == 0) break;
60         switch(menu) {
61             case 1:
62                 printf("■ 인큐 정수 데이터: ");
63                 scanf("%d", &x);
64                 if(Enqueue(&que,x) == -1) printf("Error: 인큐 실패");
65                 break;
66             case 2:
67                 if(Deque(&que, &x) == -1) printf("Error: 디큐 실패");
68                 else printf("■ 디큐 데이터: %d\n",x);
69                 break;
70             case 3:
71                 Print(&que);
72                 break;
73         }
74     }
75     Terminate(&que);
76 }
```

포인터를 이용한 연결 리스트

# 연결리스트(1)

- 연결리스트



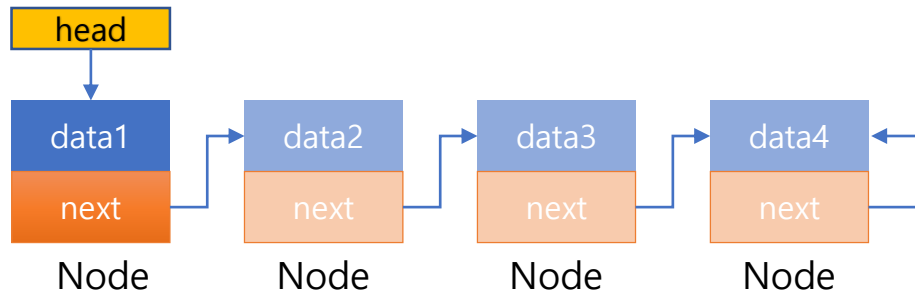
- 연결리스트가 비어있는지 판단 방법은?

`list->head == NULL`

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 typedef struct {
4     int no;
5     char name[20];
6 } Member;
7 typedef struct __node{
8     Member data;
9     struct __node *next; // 자기 참조형(self referential)
10 } Node;
11 // 멤버 추가
12 Member AddMember(const char *msg){
13     Member tmp;
14     printf("%s 멤버 입력\n", msg);
15     printf("o 번호: ");scanf("%d",&tmp.no);
16     printf("o 이름: ");scanf("%s",tmp.name);
17     return tmp;
18 }
19 //연결리스트
20 typedef struct {
21     Node *head;
22     Node *cur;
23 } List;
24 //노드를 동적으로 생성
25 static Node *AllocNode(void){
26     return (Node *)calloc(1, sizeof(Node));
27 }
28 //연결리스트 초기화
29 void Initialize(List *list){
30     list->head = NULL;
31     list->cur = NULL;
32 }
33 static void SetNode(Node *n, Member *x, Node *next){
34     n->data = *x;
35     n->next = next;
36 }
```

# 연결리스트(2)

- 연결리스트



- 노드가 1개인 연결리스트 판단 방법은?

`list->head->next == NULL`

- 노드가 2개인 연결리스트 판단 방법은?

`list->head->next->next == NULL`

- 포인터(Node \*p)가 머리 노드인지 판단 방법은?

`p == list->head`

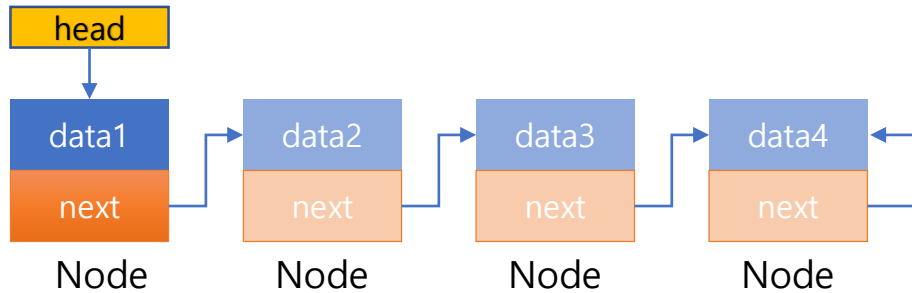
- 포인터(Node \*p)가 꼬리 노드인지 판단 방법은?

`p->next == NULL`

```
37 //현재노드앞에 노드추가
38 void InsertFront(List *list, Member *x){
39     Node *ptr = list->head;
40     list->head = list->cur = AllocNode();
41     SetNode(list->head, x, ptr);
42 }
43 //꼬리에 노드 추가
44 void InsertRear(List *list, Member *x){
45     if(list->head == NULL) InsertFront(list,x);
46     else{
47         Node *ptr = list->head;
48         while(ptr->next != NULL) ptr = ptr->next; //맨 뒤로 이동
49         ptr->next = list->cur = AllocNode();
50         SetNode(ptr->next, x, NULL);
51     }
52 }
53 //출력함수
54 void Print(List *list){
55     if(list->head == NULL){
56         printf("노드 없음");
57     }else{
58         Node *ptr = list->head;
59         printf("■ 모두 보기\n");
60         while(ptr != NULL){
61             printf("%d : %s\n", ptr->data.no, ptr->data.name);
62             ptr = ptr->next;
63         }
64     }
65 }
66 void Terminate(List *list){
67     while(list->head != NULL){
68         Node *ptr = list->head->next;
69         free(list->head);
70         list->head = list->cur = ptr;
71     }
72 }
```

# 연결리스트(3)

- 연결리스트



```
73 //Start Function
74 int main(){
75     List list;
76     Initialize(&list);
77     Member x;
78     //첫번째 멤버 입력
79     x = AddMember("첫번째");
80     InsertFront(&list, &x);
81     //두번째 멤버 입력
82     x = AddMember("두번째");
83     InsertRear(&list, &x);
84     //세번째 멤버 입력
85     x = AddMember("세번째");
86     InsertRear(&list, &x);
87     Print(&list);
88     Terminate(&list);
89 }
```

첫번째 멤버 입력

○ 번호:

1

○ 이름:

kim

두번째 멤버 입력

○ 번호:

2

○ 이름:

hong

세번째 멤버 입력

○ 번호:

3

○ 이름:

kang

■ 모두 보기

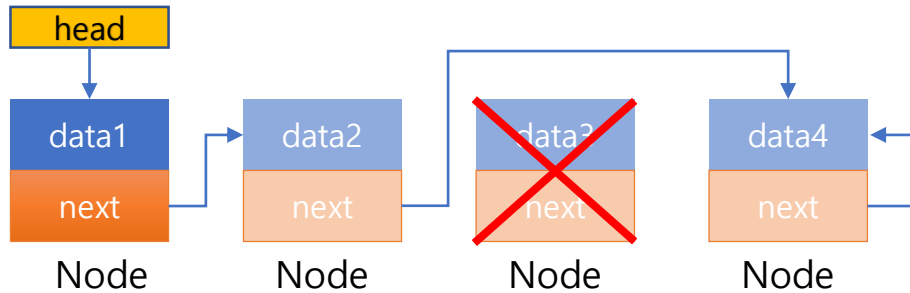
1 : kim

2 : hong

3 : kang

# 연결리스트(4)

- 연결리스트 :: 삭제



1. 검색 대상 찾기
2. 맨앞 > 맨뒤 > 중간 노드 연결 상태
3. 대상 삭제

//회원 삭제

```
Member mem = ScanMember("삭제");
RemoveMember(&list, &mem);
Print(&list);
```

```
60 // 회원노드 검색 함수
61 Node *search(List *list, Member *x){
62     Node *ptr = list->head;
63     while(ptr != NULL){
64         if(ptr->data.no == x->no && strcmp(ptr->data.name,x->name) == 0){
65             list->cur = ptr; //이전 노드
66             printf("[search]%d : %s\n", ptr->data.no, ptr->data.name);
67             return ptr;
68         }
69         ptr = ptr->next;
70     }
71     return NULL;
72 }
73 // 선택 노드 삭제
74 void RemoveMember(List *list, Member *x){
75     Node *ptr = search(list, x);
76     if(ptr != NULL && ptr == list->head){ // 헤더 노드 삭제
77         Node *dnode = list->head->next;
78         printf("[remove1]%d : %s\n", ptr->data.no, ptr->data.name);
79         free(list->head);
80         list->head = list->cur = dnode;
81     }else if(ptr->next == NULL){ //마지막노드 삭제
82         Node *pend = list->head;
83         Node *pre;
84         while(pend->next != NULL){
85             pre = pend;
86             pend = pend->next;
87         }
88         pre->next = NULL;
89         printf("[remove3]%d : %s\n", ptr->data.no, ptr->data.name);
90         free(ptr);
91         list->cur = pre;
92     }else{ // current 노드 삭제
93         Node *p = list->head;
94         Node *pre;
95         while(p != list->cur){
96             pre = p;
97             p = p->next;
98         }
99         pre->next = list->cur->next;
100        printf("[remove2]%d : %s\n", ptr->data.no, ptr->data.name);
101        free(list->cur);
102        list->cur = pre;
103    }
104 }
```