ME454
Dynamics System Programming

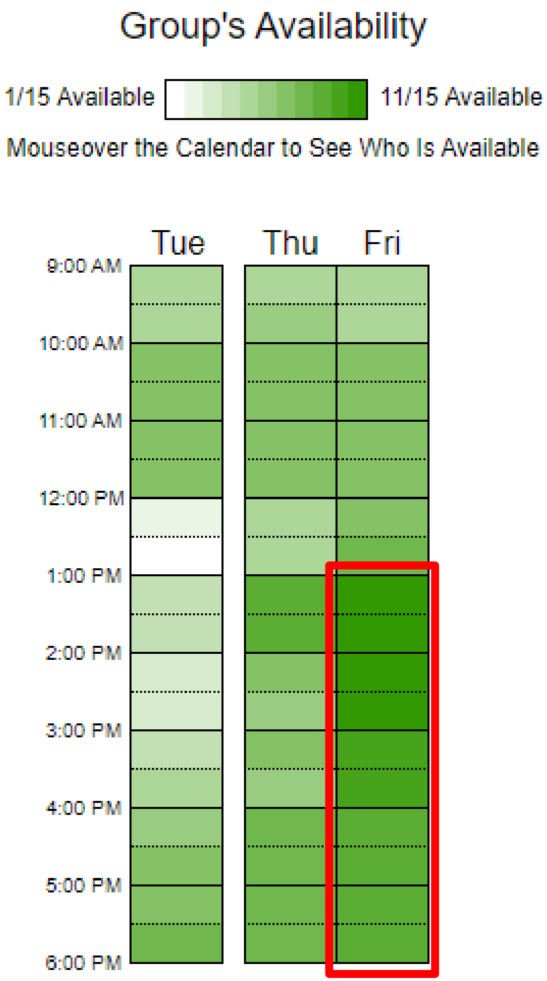
TA Session 3. C++ Basic (1)

Notice

- Homework 1 is announced on Monday
  - Due : 2024-03-18 (Mon), 23:59 PM
  - TA in charge: Minseung Kim
    - minseung_k@kaist.ac.kr

- Practice room open time
  - Friday 13:00 – 18:00
  - Staff in charge: Jungman Joo
    - 042-350-3009

- If you have any questions please ask TAs after the session.



Group's Availability

Mouseover the Calendar to See Who Is Available

Class Objectives

- Building and running a C++ program
- C++ practice (flow control, function, struct, array)
    - List implementation I (dynamic array)
- Dynamics simulation practice
    - Single rigid body object
- Supplementary materials

Directory Structure

Please make highlighted directories with `mkdir` if it is not on your computer.

Please check if your files are properly set as follows.

You may copy the files using filesystem GUI (Nautilus, Windows, …) instead of cp.

```
your home directory
```

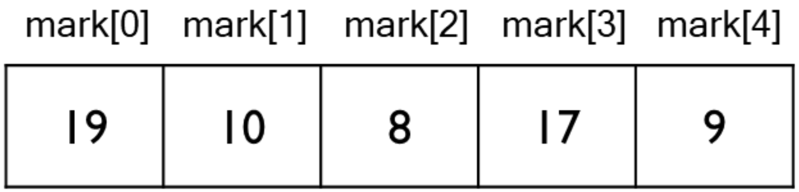
- repos
  - ME454_2024
- cpp_ws
  - cpp_practice1
- ros2_ws
  - src
    - ball_throwing
      - launch
      - worlds
      - CmakeLists.txt
      - packages.xml

```
In Ubuntu command prompt

cd ~/repos
// Downloads full repository
git clone https://github.com/skoo1/ME454_2024

// Copies C++ practice files
cp -r ~/repos/ME454_2024/cpp_practice1 ~/cpp_ws

// Copies ball_throwing practice package
cp -r ~/repos/ME454_2024/ball_throwing ~/ros2_ws/src
```

Building and Running a C++ Program

Array

- Collection of elements
  - int, float, char, struct, …
- Fixed memory space
  - Can't be longer or shorter
- Array will be introduced in more detail in next week's lecture

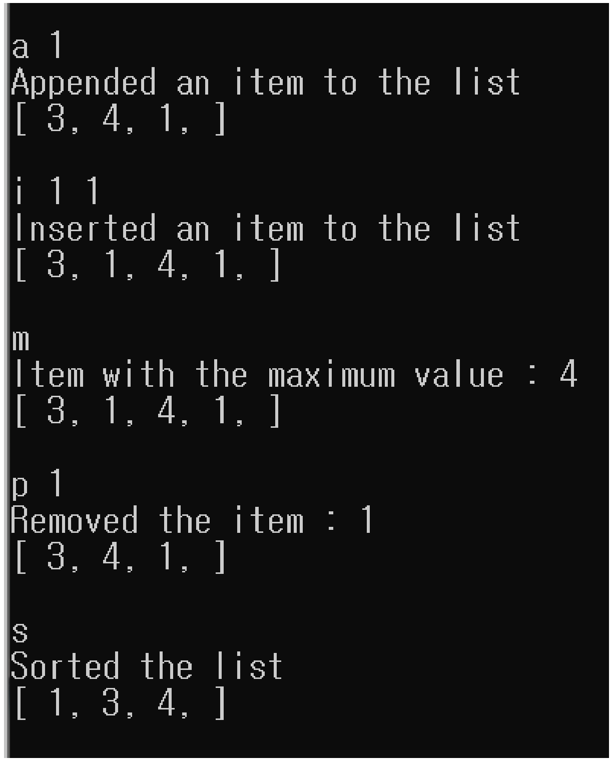


mark[0]	mark[1]	mark[2]	mark[3]	mark[4]
19	10	8	17	9

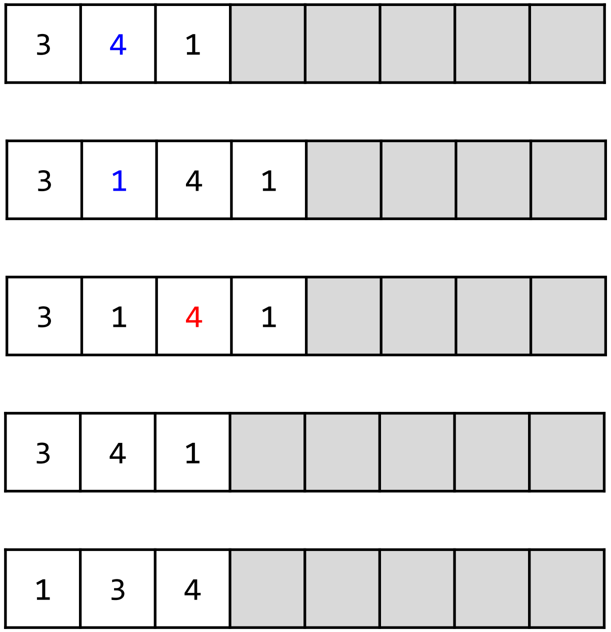
```
// initialization
int mark[5] = {19, 10, 8, 17, 9}

// print 4th element of the array
// result : 8
cout << mark[3];

// change 4th element to 9
mark[3] = 9;
```

Example Program: list4arr

- Let’s implement a list (dynamic array) that can be applied on C++ array
  - Append, Insert, Pop, Max



```
a 1
Appended an item to the list
[ 3, 4, 1, ]

i 1 1
Inserted an item to the list
[ 3, 1, 4, 1, ]

m
Item with the maximum value : 4
[ 3, 1, 4, 1, ]

p 1
Removed the item : 1
[ 3, 4, 1, ]

s
Sorted the list
[ 1, 3, 4, ]
```



3 4 1

3 1 4 1

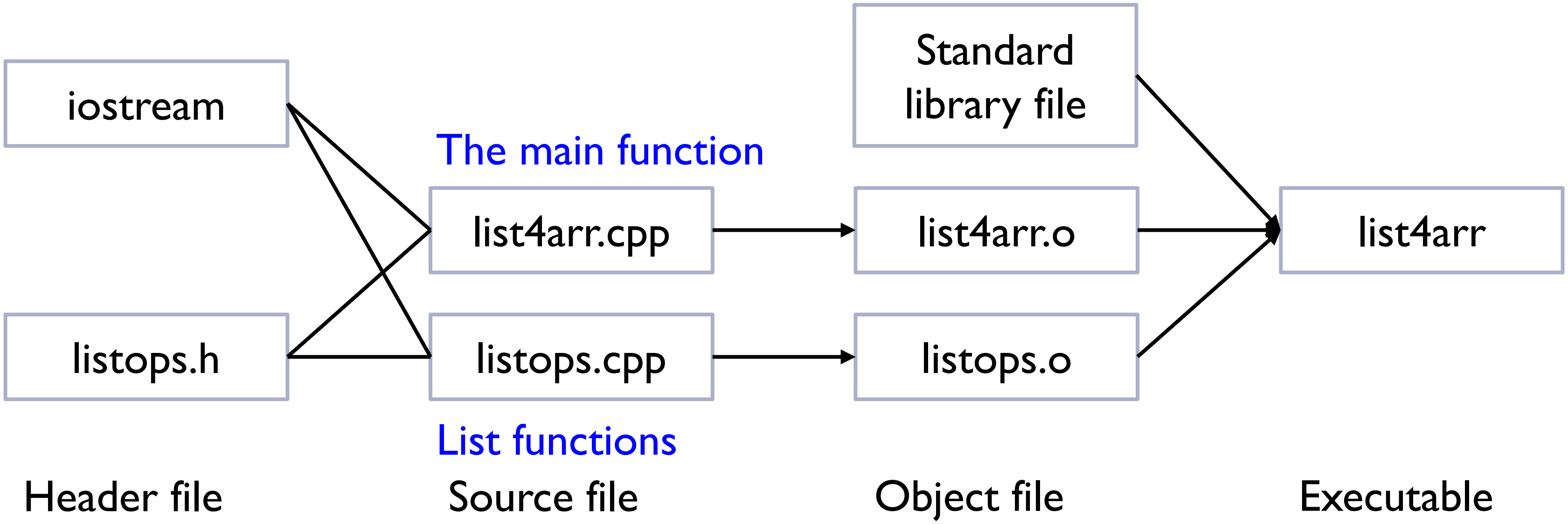
3 1 4 1

3 4 1

1 3 4

Files for list4arr

- Let's implement a list (dynamic array) with a C++ array
  - list4arr.cpp
  - listops.cpp

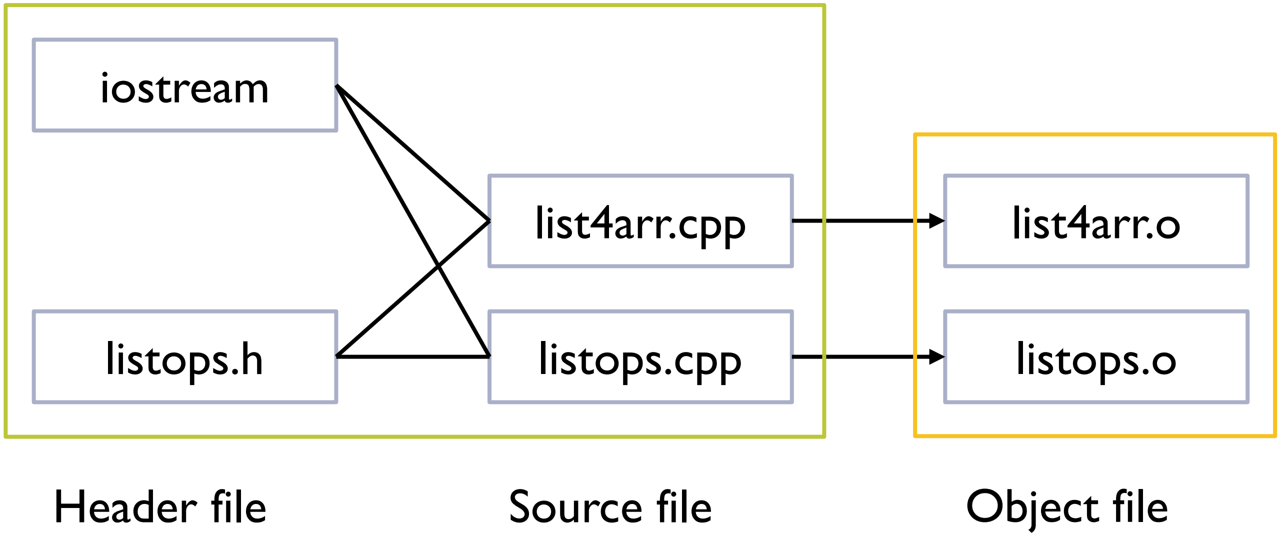


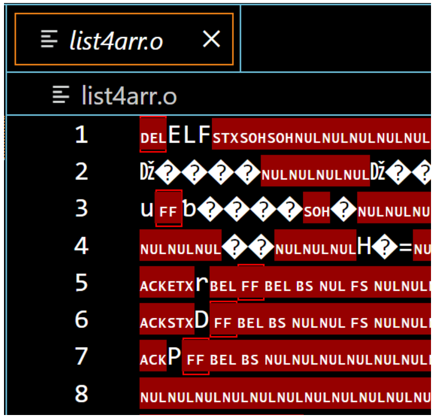
Compiler

- A compiler is a translator from code to code.
    - Human-readable code to computer-readable code

What programmers can read
(text file)

What computers can read
(binary file)





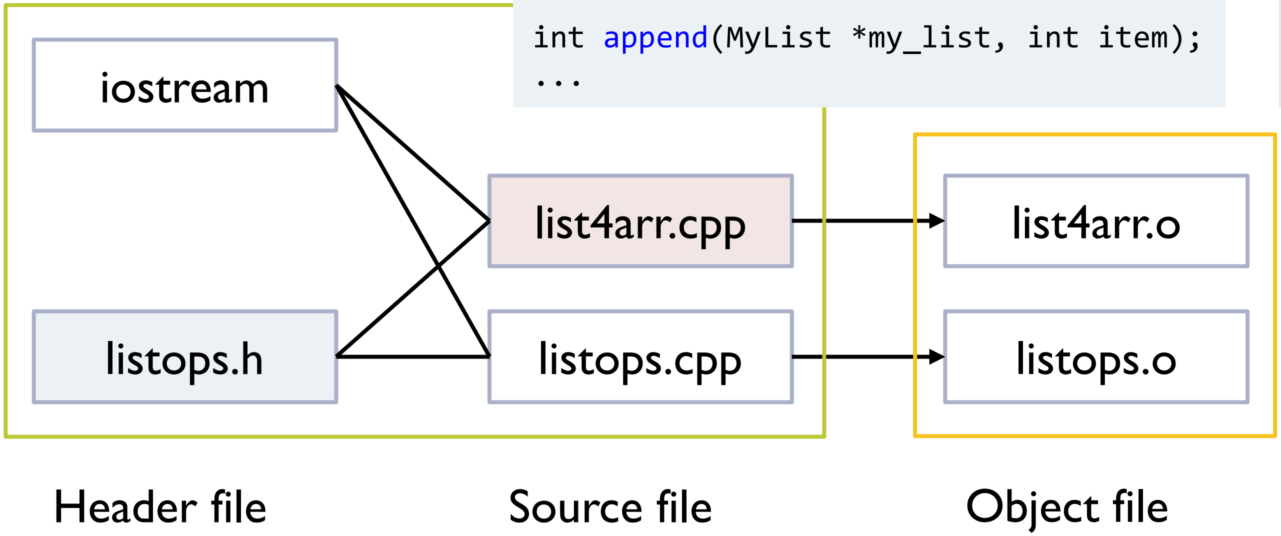
```
DELELFSTXSOHSOHNULNULNULNULNUL
Dž����NULNULNULNULDž��
uFFb����SOH�NULNULNU
NULNULNUL��NULNULNULH�=NU
ACKETXrBEL FF BEL BS NUL FS NULNUL
ACKSTXD FF BEL BS NULNUL FS NULNUL
ACKP FF BEL BS NULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNU
```

Header File

- Header files provide information to the compiler.
  - Names and types for functions, structs, objects, …

```
listops.h

struct MyList
{
    int len;
    int items[64];
};

int append(MyList *my_list, int item);
...
```

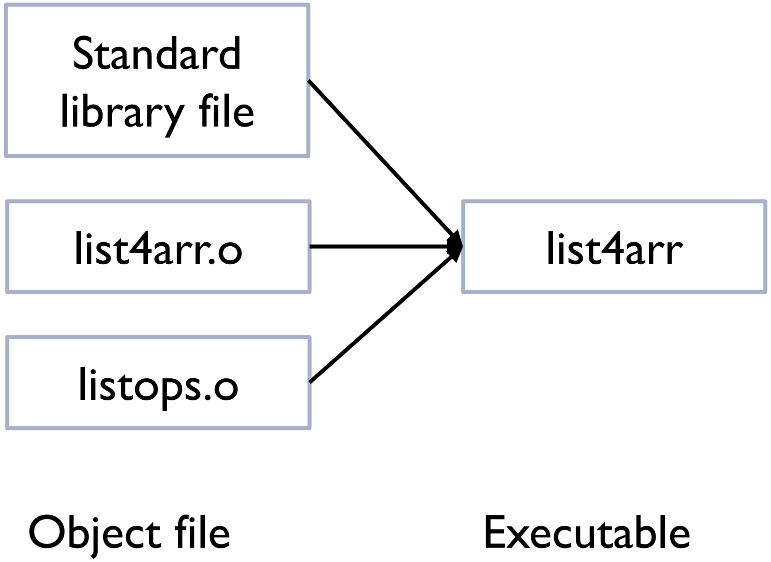
```
list4arr.cpp

#include "listops.h"

int main()
{
    MyList my_list;
    my_list.len = 0;
    append(&my_list, 3);
...
```



Linker and Build

- A linker combines one or more object files into a single executive file.
- The build process includes compiling and linking.
- You can run cmake commands on the next page instead of the commands below.

```
// GNU compiler commands to build executive

// Compile each source file
g++ -c -o listops.o listops.cpp
g++ -c -o list4arr.o lis4arr.cpp

// Link the objective files
g++ -o list4arr listops.o list4arr.o

// Clean up the objective files
rm listops.o
rm list4arr.o
```

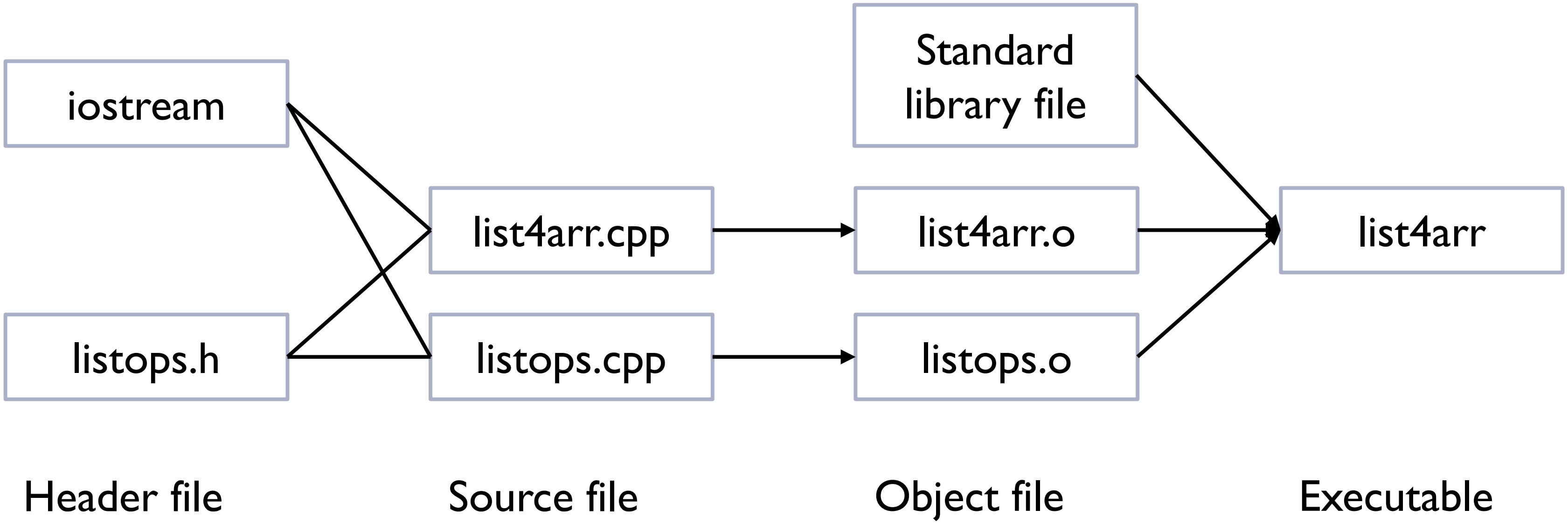


CMake

- CMake is software for the automated building.

```
// CMake commands to build executive
// In the directory with CMakeLists.txt

cmake .
make
```

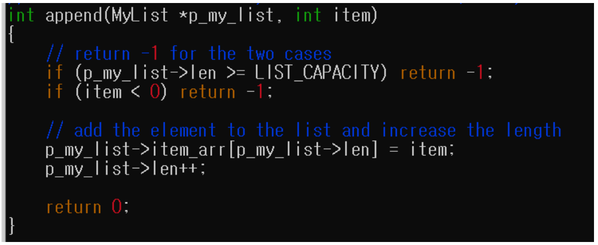
```
CMakeLists.txt

cmake_minimum_required (VERSION 3.16)
project(list4arr)
add_executable(list4arr src/listops.cpp list4arr.cpp)
```



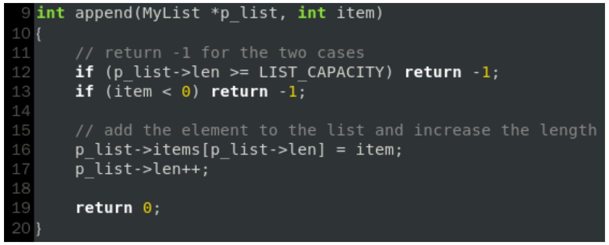
C++ Practice:
List Implementation (Dynamic Array)

Text editor

- Vim and gedit text editor can be used



```
int append(MyList *p_my_list, int item)
{
    // return -1 for the two cases
    if (p_my_list->len >= LIST_CAPACITY) return -1;
    if (item < 0) return -1;

    // add the element to the list and increase the length
    p_my_list->item_arr[p_my_list->len] = item;
    p_my_list->len++;

    return 0;
}
```

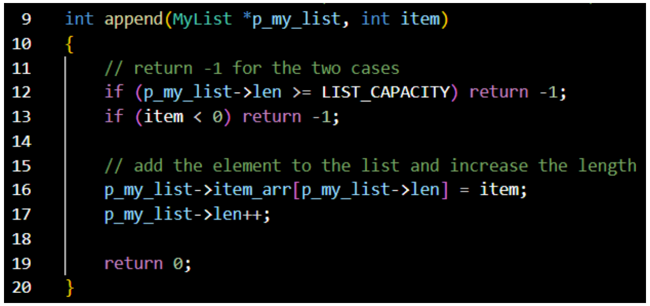


```
int append(MyList *p_list, int item)
{
    // return -1 for the two cases
    if (p_list->len >= LIST_CAPACITY) return -1;
    if (item < 0) return -1;

    // add the element to the list and increase the length
    p_list->items[p_list->len] = item;
    p_list->len++;

    return 0;
}
```

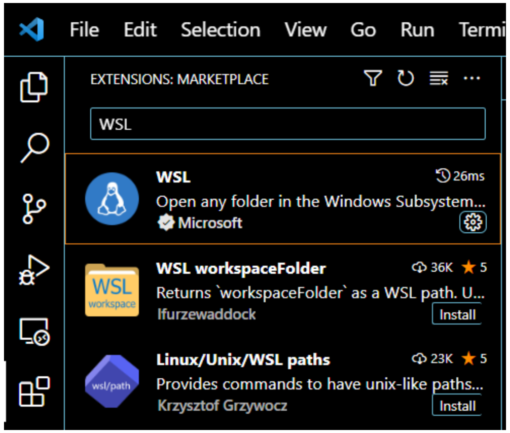
```
// Open file
vim src/listops.cpp
gedit src/listops.cpp

// Vim commands
a – insert
:q – exit | :w – save
:wq – save & exit
```

- To use Visual Studio Code in Windows
  - Download in https://code.visualstudio.com/
  - Open VSCode >> left side bar >> extensions >> WSL

```
In Ubuntu command prompt

code .
```



```
int append(MyList *p_my_list, int item)
{
    // return -1 for the two cases
    if (p_my_list->len >= LIST_CAPACITY) return -1;
    if (item < 0) return -1;

    // add the element to the list and increase the length
    p_my_list->item_arr[p_my_list->len] = item;
    p_my_list->len++;

    return 0;
}
```



WSL

WSL 26ms
Open any folder in the Windows Subsystem...
Microsoft

WSL workspaceFolder 36K ★ 5
Returns `workspaceFolder` as a WSL path. U...
lfurzewaddock Install

Linux/Unix/WSL paths 23K ★ 5
Provides commands to have unix-like paths...
Krzysztof Grzywocz Install

List Operations

- Append: adds an item to the end of the list (already implemented)
  - `append(&my_list, 1)`

3	4	1					

- Insert: adds an item to the specific position (index) of the list
  - `insert(&my_list, 1, 1)`

3	1	4	1				

- Max: returns the biggest item in the list
  - `max(&my_list)`

3	1	4	1				

- Pop: removes and returns an item from the specific position (index) of the list
  - `pop(&my_list, 1)`

3	4	1					

```
listops.h

int append(MyList *p_list, int item);
int insert(MyList *p_list, int item, int index);
int max(MyList *p_list);
int pop(MyList *p_list, int index);
```

Structure and Functions

- Structure MyList has the list length and array to store non-negative integer items.
- In the list operation functions, you can access members in MyList with (`p_list->len`) instead of (`p_list.len`).
  - The reason (pointer) will be explained in the next week's lecture
- The return statement terminates the function.

```
listops.h

#define LIST_CAPACITY 64

struct MyList
{
    int len;
    int items[LIST_CAPACITY];
};
```

```
len: 5
items:
```

3	1	4	1	5			

```
listops.cpp

int append(MyList *p_list, int item)
{
    // return -1 for the two cases
    if (p_list->len >= LIST_CAPACITY) return -1;
    if (item < 0) return -1;

    // add the element to the list and increase the length
    p_list->items[p_list->len] = item;
    p_list->len++;

    return 0;
}
```

Other Notices

- The contents of MyList should be non-negative integers:
  - Ranging from 0 to 2,147,483,647 (maximum value of 32-bit int)

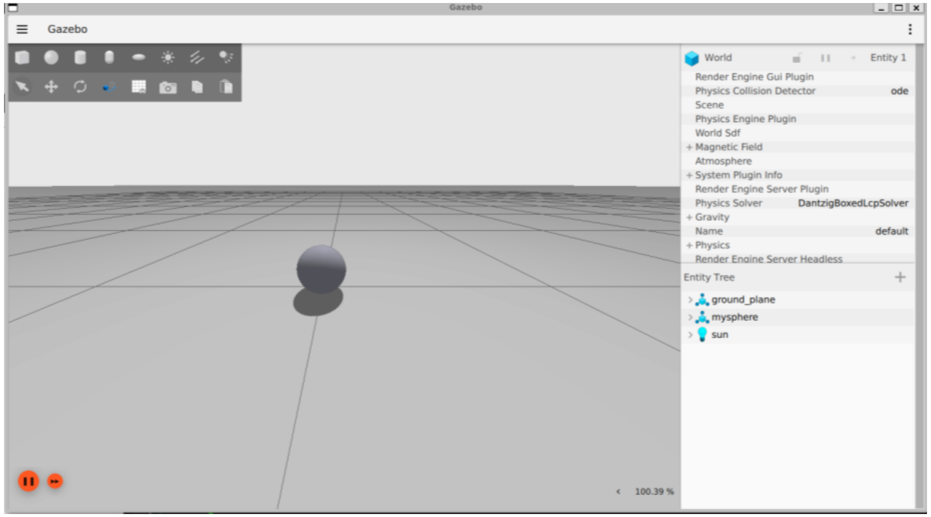
- With the Boolean variable `interact`, you can select from two modes
  - User interaction mode: demonstration for standard console input and output
  - Pre-set command mode: may be modified to test the functions

- TAs will check if the functions are implemented with the evaluation code.
  - `./list4arr_eval`
  - It is okay to refer to the source `list4arr_eval.cpp`, but please don't modify it.
  - Don't forget to build the program using `make` after code modification.
  - Please read carefully the description comment in the source code.

Dynamics Simulation Practice:
Single Rigid Body Object

World File for Ball Throwing

- World file `worlds/ball.sdf` is in SDF (simulation description format)
  - Physics such as `<gravity>`
  - Objects as `<model>`
    - Inertial properties `<inertia>` : mass, moment of inertia
    - Visual properties `<visual>` : geometry, color
    - Collision properties `<collision>` : geometry



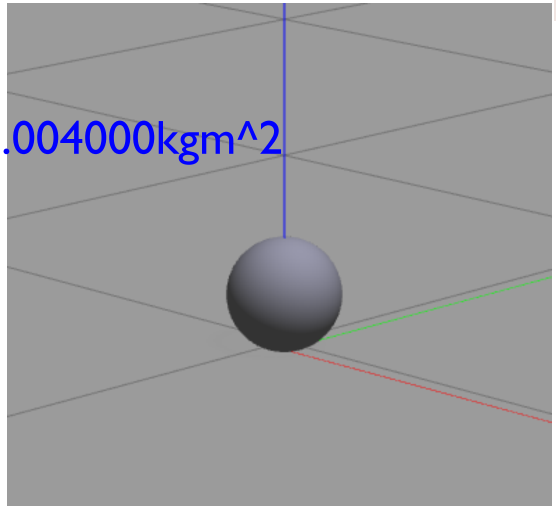
Render Engine Gui Plugin
Physics Collision Detector ode
Scene
Physics Engine Plugin
World Sdf
+ Magnetic Field
Atmosphere
+ System Plugin Info
Render Engine Server Plugin
Physics Solver DantzigBoxedLcpSolver
+ Gravity
Name default
+ Physics
Render Engine Server Headless

Sphere and Cylinder Geometry

- Specified in the world file

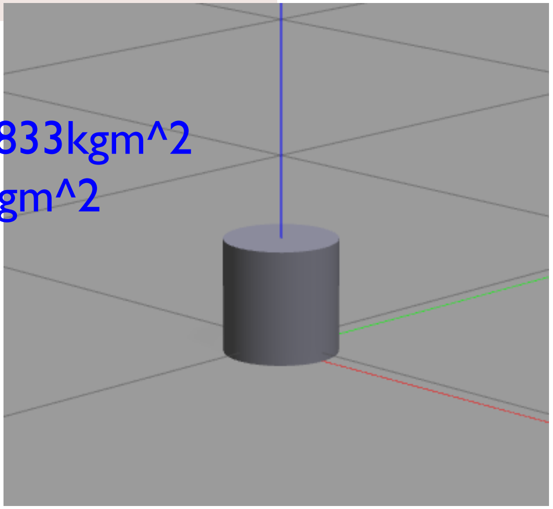
```
// SDF description for Sphere

<geometry>
  <sphere>
    <radius>0.1</radius>
  </sphere>
</geometry>
```

M = 1.0kg
Ix = Iy = Iz = 0.004000kgm^2



```
// SDF description for Cylinder

<geometry>
  <cylinder>
    <radius>0.1</radius>
    <length>0.2</length>
  </cylinder>
</geometry>
```

M = 1.0kg
Ix = Iy = 0.005833kgm^2
Iz = 0.005000kgm^2

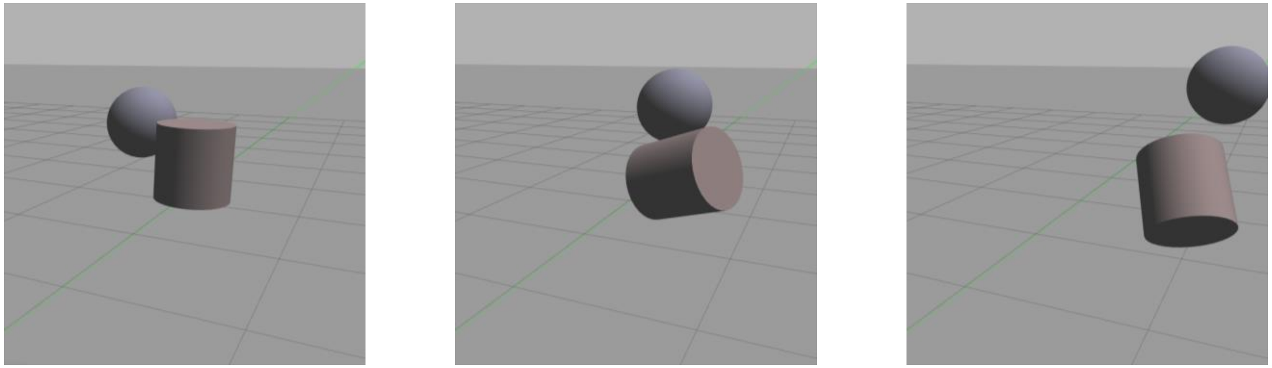


Collision Simulation

- The world file (ball_cylinder.sdf) was set as follows.
  - Set zero gravity
  - Add one sphere and one cylinder model
    - Inertia, visual, collision are as the previous slide.

```
In Ubuntu command prompt

code .
```

- You may observe collision after setting a velocity of the ball.
- Please terminate Gazebo using ctrl+c in the prompt, not X button in GUI.
  - If there is any problem running gazebo, you may kill all processes named 'ruby'.



Running Gazebo with ROS Launch

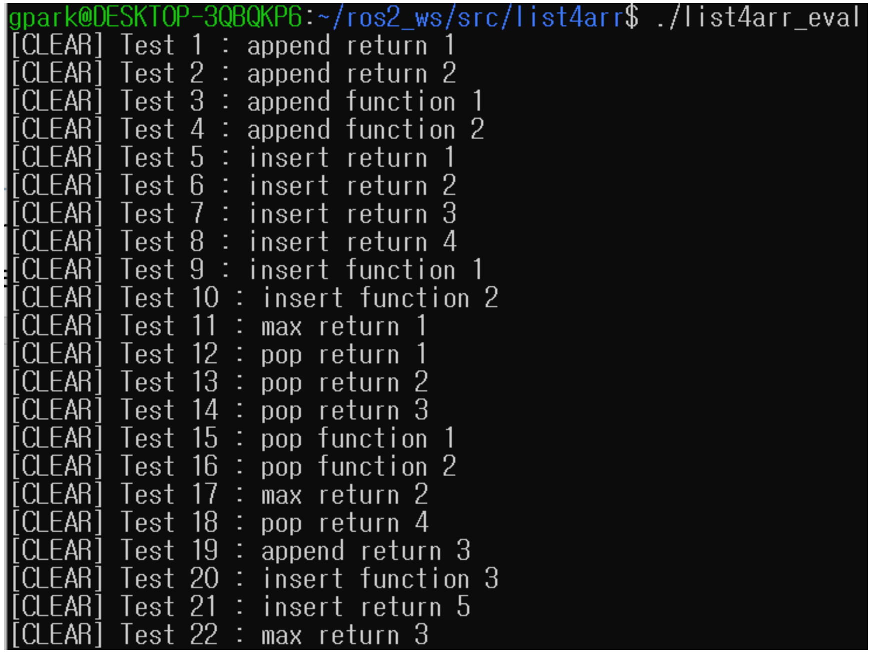
- ROS2 launch can start several executables at once.
- ROS2 topic sends data to the simulation (to be explained in week 6).

```
// Ubuntu command to start Gazebo simulation

// Install colcon (only once)
sudo apt install python3-colcon-common-extensions

cd ~/ros2_ws

// Build a ROS package and add the path to executables
colcon build
. install/setup.bash

// Launch Gazebo simulation in the package
// ros2 launch [package_name] [launch_file_name]
ros2 launch ball_throwing ball_launch.py
```

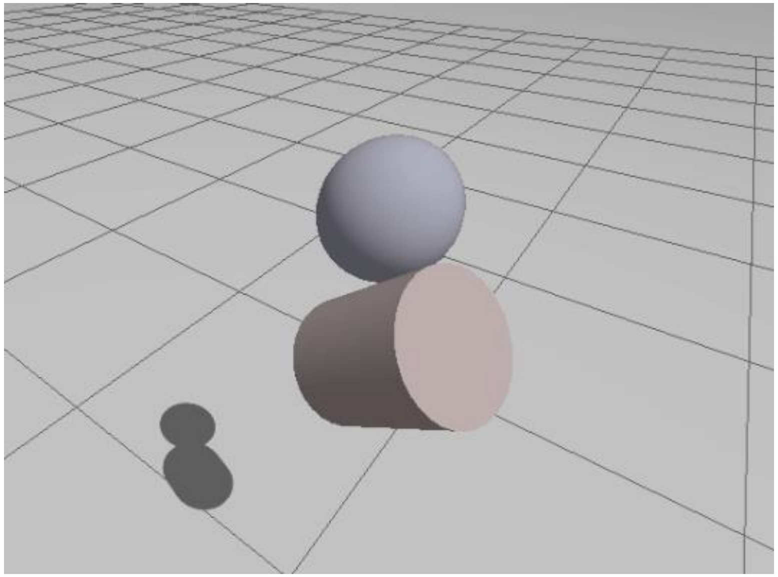
```
// Ubuntu command to set object state (In another Ubuntu terminal)

// Publish Ros2 topic to set velocity of the ball (no line separation)
ros2 topic pub /model/mysphere/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 1.0, y: 0.1, z:
0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
```

TODO

- Complete all the test in `list4arr_eval` and show TAs the screen.
- Try collision simulation of a sphere and a cylinder
  - You may capture it with screen capture tool in Windows.



```
gpark@DESKTOP-3QBQKP6:~/ros2_ws/src/list4arr$ ./list4arr_eval
[CLEAR] Test 1 : append return 1
[CLEAR] Test 2 : append return 2
[CLEAR] Test 3 : append function 1
[CLEAR] Test 4 : append function 2
[CLEAR] Test 5 : insert return 1
[CLEAR] Test 6 : insert return 2
[CLEAR] Test 7 : insert return 3
[CLEAR] Test 8 : insert return 4
[CLEAR] Test 9 : insert function 1
[CLEAR] Test 10 : insert function 2
[CLEAR] Test 11 : max return 1
[CLEAR] Test 12 : pop return 1
[CLEAR] Test 13 : pop return 2
[CLEAR] Test 14 : pop return 3
[CLEAR] Test 15 : pop function 1
[CLEAR] Test 16 : pop function 2
[CLEAR] Test 17 : max return 2
[CLEAR] Test 18 : pop return 4
[CLEAR] Test 19 : append return 3
[CLEAR] Test 20 : insert function 3
[CLEAR] Test 21 : insert return 5
[CLEAR] Test 22 : max return 3
```



TIPS

- Please double check page 5.
- Please use commands in the text file in KLMS if you would copy and paste them.
- If you have a graphics problem, try rebooting your WSL.

```
In Ubuntu command prompt

sudo shutdown now
```

- You need to build a program again every time the source code changes.
  - Use make commands on page 14 to build C++ programs.
  - Use the commands on page 24 to build ROS packages.

- We recommend putting the following in .bashrc, if you didn’t.

```
// in .bashrc

export LIBGL_ALWAYS_SOFTWARE=1
source /opt/ros/humble/setup.bash
```