Memory Management

HoangND1

Objectives

1. Casting
2. C Program structure in memory
3. Allocate dynamic memory
4. Manipulate in memory

Type casting

- All objects in C have specified type
  - ✓ Type variable char, int, float, double, …
  - ✓ Pointers point to type char, int, float, double, …
- Expression with many types
  - ✓ C language automatic cast the types (casting).
  - ✓ User cast the types.

Implicit casting

- Increase level (data type) in expression
  - ✓ Elements with the same type
    - The result is general type
    - int / int ➔ int, float / float ➔ float
    - Example: 2 / 4 ➔ 0, 2.0 / 4.0 ➔ 0.5
  - ✓ Elements with the diffirent type
    - The result is cover type
    - char < int < long < float < double
    - float / int ➔ float / float, …
    - Example: 2.0 / 4 ➔ 2.0 / 4.0 ➔ 0.5
    - Note: temporary casting

Implicit casting

- Assign <left expression> = <right expression>;
  - ✓ The right expression is increased level (or reduced level) temporary as the same type with right expression type.

```
int i;
float f = 1.23;

i = f;       // ➔ f temporary is int
f = i;       // ➔ i temporary is float
```

- ✓ May be the accurate of real will be lost ➔ limited!

```
int i = 3;
float f;
f = i;       // ➔ f = 2.999995
```

Explicit casting

- Meaning
  - ✓ Type casting to avoid wrong result.
- Syntax

```
(<new type>)<expression>
```

- Example

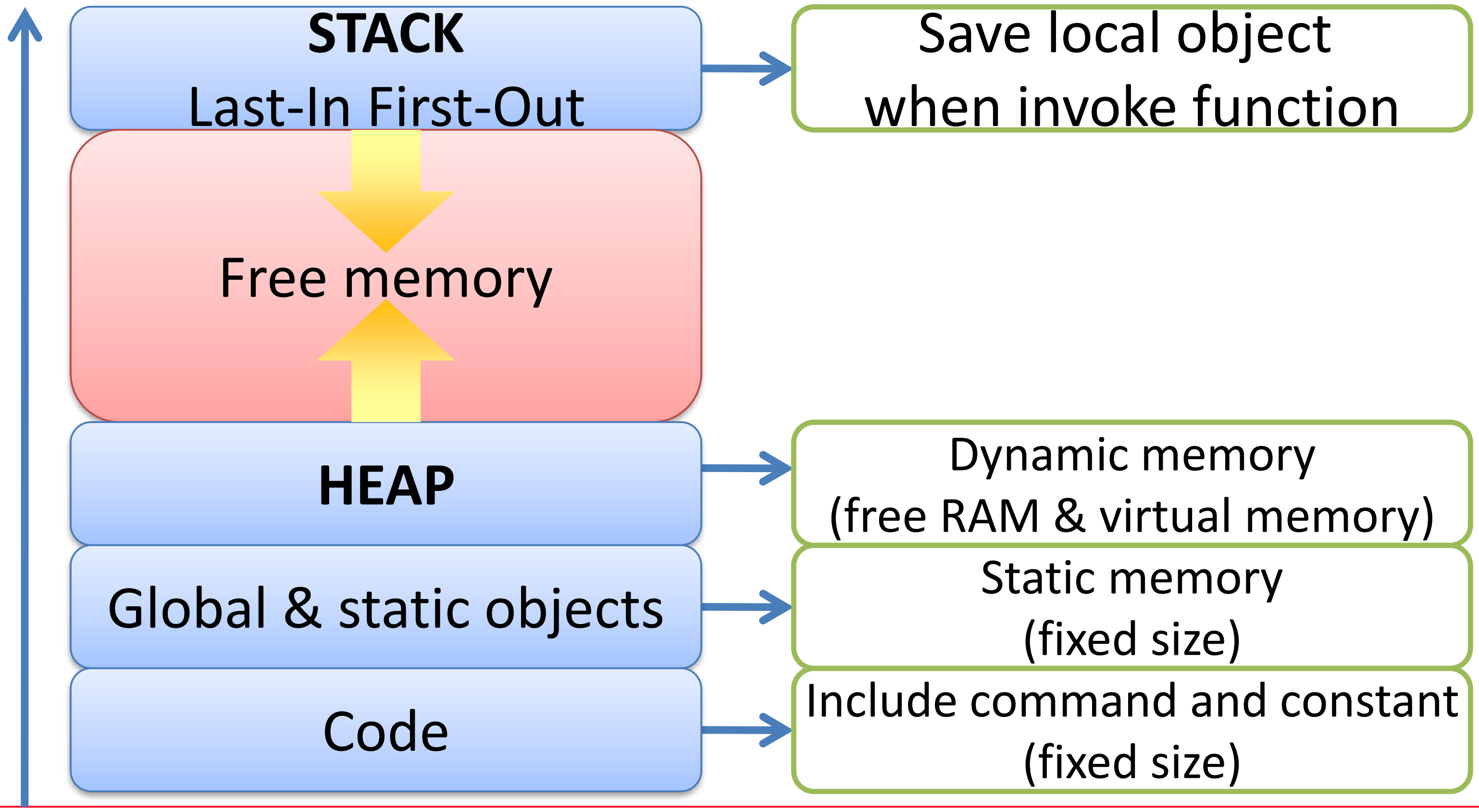
```
int x1 = 1, x2 = 2;
float f1 = x1 / x2;             // ➔ f1 = 0.0
float f2 = (float)x1 / x2;      // ➔ f2 = 0.5
float f3 = (float)(x1 / x2);    // ➔ f3 = 0.0
```

Allocate static and dynamic memory

- Static memory allocation
  - ✓ Declare variable, struct, array …
  - ✓ Must know how many memories to store ➔ waste memory, can not change size, …
- Dynamic memory allocation
  - ✓ Allocate as required.
  - ✓ Free the memory if not need.
  - ✓ Use outside memory (include virtual memory).

C program structure in memory

- The whole of program will be loaded into memory which is free, with 4 parts:

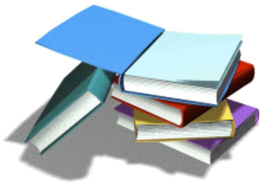

STACK
Last-In First-Out
Save local object when invoke function
Free memory
HEAP
Dynamic memory (free RAM & virtual memory)
Global & static objects
Static memory (fixed size)
Code
Include command and constant (fixed size)

Allocate dynamic memory

- Library <stdlib.h> or <alloc.h>
  - ✓ malloc
  - ✓ calloc
  - ✓ realloc
  - ✓ free

Allocate dynamic memory

void *malloc(size_t size)



Allocate in HEAP a memory size (bytes)
size_t instead of unsigned (in <stddef.h>)

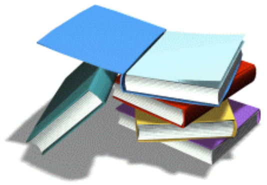
Return

- Success: The pointer point to allocated memory.
- Fail: NULL (not enough memory).

```
int *p = (int *)malloc(10*sizeof(int));
if (p == NULL)
		printf(“Not enough memory!”);
```

Allocate dynamic memory

void *calloc(size_t num, size_t size)



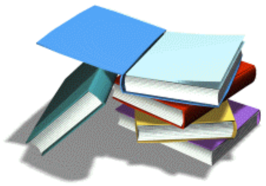
Allocate memory include num elements in HEAP, each has size (bytes)

- Success: The pointer point to allocated memory.
- Thất bại: NULL (not enough memory).

```
int *p = (int *)calloc(10, sizeof(int));
if (p == NULL)
        printf("Not enough memory!");
```

Allocate dynamic memory

void *realloc(void *block, size_t size)



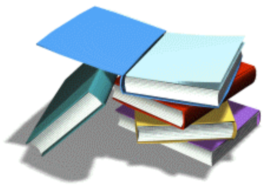
Reallocate memory with size that block point memory in HEAP.
block == NULL ➔ use malloc
size == 0 ➔ use free

- Success: The pointer point to allocated memory.
- Fail: NULL (not enough memory).

```
int *p = (int *)malloc(10*sizeof(int));
p = (int *)realloc(p, 20*sizeof(int));
if (p == NULL)
        printf("Not enough memory!");
```

Allocate dynamic memory

void free(void *ptr)



Free memory pointed by ptr, that returned by malloc(), calloc(), realloc() functions.
If ptr is NULL -> do nothing.

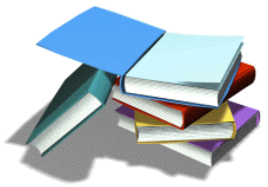
- Nothing.

```
int *p = (int *)malloc(10*sizeof(int));
free(p);
```

Fpt Software

Allocate dynamic memory

<pointer_to_datatype> = new <datatype>[size]



Allocate memory with size = sizeof (<datatype>)* in HEAP

Return

◆ Success: The pointer point to allocated memory.
◆ Fai: NULL (not enough memory).

```
int *a1 = (int *)malloc(sizeof(int));
int *a2 = new int;
int *p1 = (int *)malloc(10*sizeof(int));
int *p2 = new int[10];
```

Allocate dynamic memory

delete []<pointer_to_datatype>

Free the memory in HEAP pointed by <pointer_to_datatype> (allocated by new)

- Nothing.

```
int *a = new int;
delete a;
int *p = new int[10];
delete []p;
```

Allocate dynamic memory

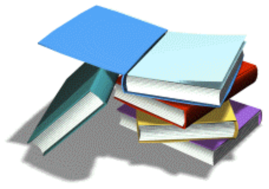
- Note
  - ✓ Not need check the pointer is NULL or not before free or delete.
  - ✓ Allocate by malloc, calloc or realloc -> free the memory by free.

Manipulate in memory

- Library <string.h>
  - ✓ memset : assign value to all bytes in memory.
  - ✓ memcpy : copy memory.
  - ✓ memmove : move information from memory to memory.

Manipulate in memory

void *memset(void *dest, int c, size_t count)



Assign first count (bytes) of memory pointed by dest with value c (from 0 to 255)
Use for char memory, with other type memory -> the value is zero .

- pointer dest.

```
char buffer[] = "Hello world";
printf("Before memset: %s\n", buffer);
memset(buffer, '*', strlen(buffer));
printf("After memset: %s\n", buffer);
```

Manipulate in memory

void *memcpy(void *dest, void *src, size_t count)

Copy count byte from src memory into dest memory.
If 2 memories overlap, the function works **not** exactly.

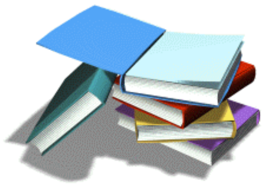
Return

- Pointer dest.

```
char src[] = "*****";
char dest[] = "0123456789";
memcpy(dest, src, 5);
memcpy(dest + 3, dest + 2, 5);
```

Manipulate in memory

void *memmove(void *dest, void *src, size_t count)



Copy count byte from src memory into dest memory.
If 2 memories overlap, the function works exactly.

- Pointer dest.

```
char src[] = "*****";
char dest[] = "0123456789";
memmove(dest, src, 5);
memmove(dest + 3, dest + 2, 5);
```

Questions and Answers