CS 231 Title

Jnotes

Basic Data Type
↳ int, float, char, bool, string

Abstract Data Type
↳ stack, queue, list ← linear data structure

↳ tree, graph ← non-linear

Binary Number Representation
int → unsigned
$(1011)_2 = 11$

→ signed magnitude
$(1011)_2 = -3$
0 = +
1 = -

→ 1's complement
$(1011)_2 = -4$
0 = +
1 = -
↳ $(0100)_2$

→ 2's complement
$(1011)_2 = -5$
0 = +
1 = -
↳ $(0100)_2 + 1$
4 + 1

Example: Basic Data Type
int → จำนวนเต็ม
float → จำนวนทศนิยม
bool → true, false

Overflow → เกิน
char (8 bit) <-128 to 127>
125 + 10 = 135

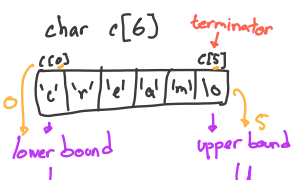
Underflow น้อยเกิน
char 8bit <-128 to 127>
-135

floating point representation
single-point precision 4 bytes <float>
↳ 32 bits 1 (s) | 8 (E) | 23 (M)
+/- $2^{\square}$.____ Bias = 127

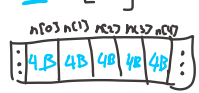
double-point precision 8 bytes <double>
↳ 64 bit 1 11 52 bias = 1023

Data Types in C
int → 4 bytes (32 bit) -2^{31} to $2^{31}-1$ %d
u_int → 4 bytes (32 bit) 0 to $2^{32}-1$ %u
char (signed) → 1 byte (8 bit) -128 to 127 %c
u_char → 1 byte (8 bit) 0 to 255 %u
short → 2 bytes (16 bit) -2^{15} to $2^{15}-1$ %hd (half)
u_short → 2 bytes (16 bits) 0 to $2^{16}-1$ %hu

long → 8 bytes (64 bits) -2^{63} to $2^{63}-1$ %ld (long)
float → 4 bytes (32 bit) %f
double → 8 bytes (64 bit) %lf (long)



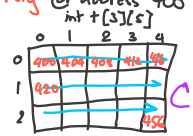
Array
int n[5]

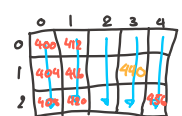


int a[25] @ adress 400
a[8] @ address 432

Loc(a[i])
= base + (w * (i - L))
400 int 4 8 0

2D array @ address 400
int +[3][5]





3D array
b[0:2,5:9,1:4]
3 x 5 x 4
= 60 members

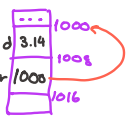
String Functions

- strcmp(char*, char*)
  - 0 → s1 == s2
  - \+ → s1 > s2
  - \- → s1 < s2
- strcpy(char* dest, char* src)
- strcat(char* dest, char* src)
- strlen(char*) ไม่นับ '\0'

Pointer

variable contain an address <uses 8 bytes> ∴ p

```
double d = 3.14;
double *ptr = &d;      → 1000
double o = *ptr;       → 3.14
```




```
int x=1, y=2;
int *ip, *iq;
ip = &x;
y = *ip;
*ip = 0;
y = 5;
ip = &y;
*ip = 3;
iq = ip;
```

Adding Pointer w/ number.

short s = 3; → @250

2 short *ptr = &s → 250

ptr += 8; → 256

Struct

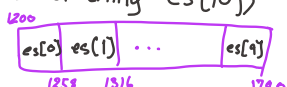
```
struct entity {
    char[50] name;
    int level;
    float hap;
};
```

58 bytes

```
struct entity oat;
```


```
oat.name;  <accessing data in a struct>
oat.level = 768;  <assigning data>
strcpy(oat.name, "Jessie");
```

```
struct entity es[10];
```




```
struct A { };
struct B { };
struct C {
    struct A a;
    struct B b;
};
```

nest structure

```
struct entity *e_ptr = es[2];   → 1316
```

<pointer to struct>

Typedef

```
typedef struct entity Entity
        <type>
```

Union

```
union smith {
    int i;
    char c;
};
union smith u;
u.i = 100;
u.c = 'A';
cout << u.c << endl;   A
cout << u.i << endl;   65
```

u.i / u.c

u: 100 'A'

u.i: 100

u.c: 'A'

Functions

```
#include <iostream>   ← import library.
```

```
type func_name(t1, t2, ..., tn);
ex: int idk(int, int);   ← function prototype.
```

Variables

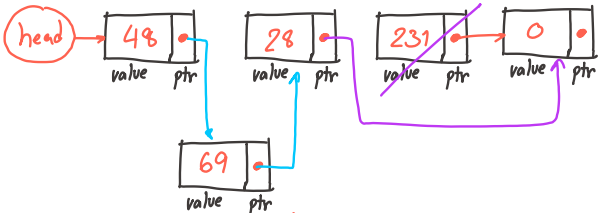
global → นอก function ใครใช้ก็ได้

local → ใน function ใครมี=function

pass by value → ส่งค่า ex: func(int)

pass by reference → ส่ง address ex: func2(int*)

Linked List



node

```
typedef struct _node{
    int value;
    struct _node *next;
} node;
```

heap memory

linked list types

Stack

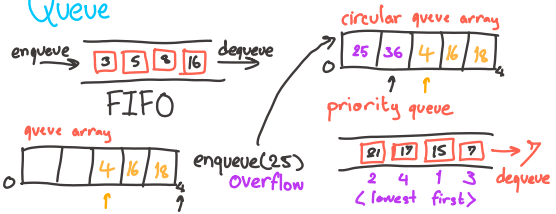
Infix, prefix, postfix

A+B-C -+ABC AB+C-

() > ^ > x/ > +-

ex: A+BxC-D ABCx+D-

Queue



Recursion calls itself

```
int f(int n) {
    if (n <= 0) {return 0;}
    else {return f(n-1) + n;}
```

recursion

linear recursion

f(3) = f(2) + 3 = 3+3 = 6

f(2) = f(1) + 2 = 1+2 = 3

f(1) = f(0) + 1 = 0+1 = 1

f(0) = 0

tail recursion

f(3,0) (count)

f(2,0+3) (result)

f(1,3+2)

f(0,5+1) = f(0,6) return

binary recursion

(feat. fibonacci)

fi(4) = fi(3) + fi(2)

fi(2)+fi(1) fi(1)+fi(0)

fi(1)+fi(0)

multiple recursion

(feat. ackermann)

ack(1, 1)

ack(0, ack(1,0))

ack(0,1)

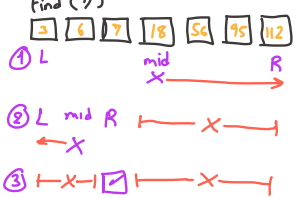
Searching

sequential search $O(n)$

binary search $O(\log_2 n)$

(must be sorted)

find (7)



hashing

1. mod

make m-size hash table

key mod m

ex: 23 mod 11 = 1

$H(K) = K \bmod m$

↑ reserved spaces

ex: 11 50 48 18 36

$H(K) = K \bmod 7$