C

=> to find/control precise no. of floating digits

```
printf("%.10f", a);
        ↳ alter this number to inc/dec.
```

=> %.P ⟶ pointers

=> bool - 1 byte
char - 1 byte
double - 8 bytes
float - 4 bytes
int - 4 bytes
long 8-bytes.

=> if any value in a calculation is a float, then the whole calculation becomes float.

arrays

=>
```
int scores[3];
scores[0] = 72;
scores[1] = 73;
```

=>
```
const int name = 4;
```

=> header files (Ex - stdio.h) to import/include libraries of C.

JUN

Notes

Appointment

04 June Wednesday

! → bang
↳ pronounced in programming

JUNE 2008

Mon	30	02	09	16	23	
Tue		03	10	17	24	
Wed		04	11	18	25	
Thu		05	12	19	26	
Fri		06	13	20	27	
Sat		07	14	21	28	
Sun	01	08	15	22	29	

=> when we store string, another character '\0' (null character) ~~to~~ figure out the length of string is stored by C.

=> uppercase.c

```
#include <stdio.h>
#include <cs50.h>
#include <string.h>
int main(void)
{ ~~for(int i=0,~~
  string s = get_string("Before: ");
  printf("After: ");

  for(int i=0, n=strlen(s); i<n; i++)
  {
      if(s[i]>='a' && s[i]<='z')
      {
          printf("%c", s[i]-32);
      }
      else
      {
          printf("%c", s[i]);
      }
  }
}
```

Notes

Appointment

manual.cs50.io

:) Command line arguments.

```
#include <stdio.h>

int main(int argc, string argv[])
{

}
```

argc → stores no. of inputs (including ./...)

argv[] → stores array of inputs {./..., x1, x2}

Types of Big O notation:-

$O(1)$, $O(n^2)$, $O(n \log n)$, $O(n)$, $O(\log n)$

↳ if there is a fixed (const.) steps.

Ω notation → refers to minimum no. of steps.

$O(n)$ $\Omega(1)$

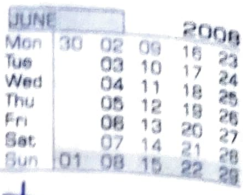
Linear search → searching one by one from 0, 1, ... n

Binary search → if nos. are sorted, go to middle & proceed accordingly. $(\Omega(1))$

Notes Appointment

06 June Friday

O, Ω, θ
Big O ↓ omega ↓ theta



=> You can do for 'int' & 'char' but you can't use "==" to compare strings in C.

=> typedef struct
{
string name;
string number;
}
person;

Brian - +16174951000
David - +19194682750

person people[2];

people[0].name = "yash";

=> selection sort - looking at each element of an array one-by one & putting smallest ~~on~~ element to the left

$$n + (n-1) + (n-2) + \cdots 1$$
$$= n(n+1)/2$$
$$= n^2/2 + n/2$$

Hence, $O(n^2)$

Notes

Appointment

=) bubble sort - comparing 2 digits & swapping

☆ pseudo code for selection sort -

for i from 0 to n-1
Find smallest item b/w ith & last item
Swap smallest item with ith item.

☆ Pseudo code for bubble sort -

Repeat (until sorted) or (n-1 times)
For i from 0 to n-2
if ith & (i+1)th items out of order
Swap them

do a function

=) Recursion → ability to call a function themselves - fact(n) = n * fact(n-1)

Sunday 08

=) merge sort (O(n log n)
pseudo code -
sort left half numbers
sort right half number
merge sorted halves

Notes

Appointment

09 June Monday

JUNE 2008

Mon	30	02	09	16	23
Tue		03	10	17	24
Wed		04	11	18	25
Thu		05	12	19	26
Fri		06	13	20	27
Sat		07	14	21	28
Sun	01	08	15	22	29

Ex Recursion factorial function

```
int fact(int n)
{
    if(n==1)
    {
        return 1;
    }
    else
    {
        return n * fact(n-1);
    }
}
```

☆ `#define MAX 9` = `const int MAX = 9;`

⇒ Hexadecimals include - 0,1,2,3,4,5,6,7,8,9
A,B,C,D,E,F

⇒ RGB - Colour code I

#XXXXXX
RR GG BB
Hexadecimals

Notes

Appointment

& → Give the memory address of
* → look at the memory address
↳ dereference operator

Pointer - A variable that contains address of some other value.

=> int *p = &n;
printf("%-p\n", p);

=) char *s = "HI!";
printf("%-c", *s);
*(s+1)
s[1]

=) Segmentation fault - to touch the memory which not belongs to you

=> char *i = "yash";
char *j = "yash";
if(i == j)
{
return 1;
}
else
return 2;

Notes

output :→ 2
Because string is the address of the 1st character.

Appointment

11 June Wednesday

JUNE 2008

Mon	30	02	09	16	23
Tue		03	10	17	24
Wed		04	11	18	25
Thu		05	12	19	26
Fri		06	13	20	27
Sat		07	14	21	28
Sun	01	08	15	22	29

pointer - 8 bytes

=> #include <string.h>
strcmp(s,t)
↳ returns '0' if strings are identical.

=> char *t = malloc(4);
↳ How many bytes of memory do you want to allocate to the variable.

#include <stdlib.h>

if (t == NULL)
{
return 1;
}

=> strcpy(t,s);

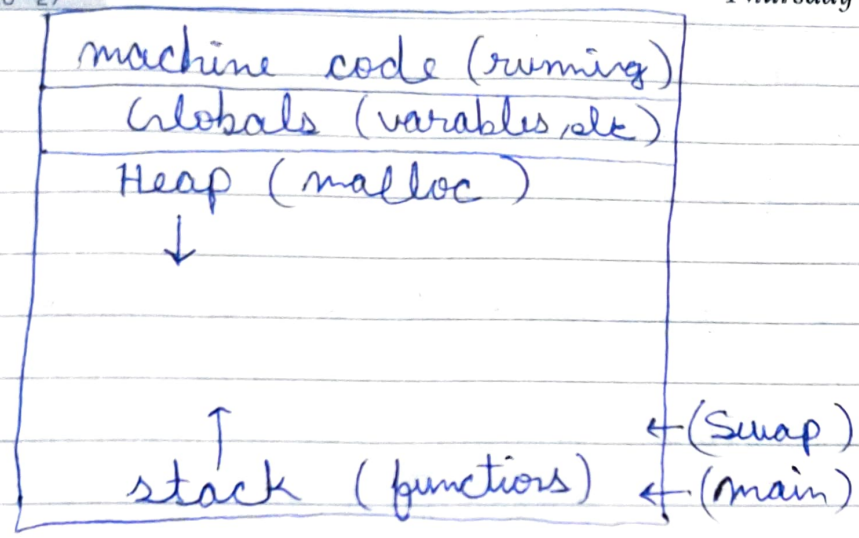
=> free(t); → to free the memory after allocating memory [URGENT]

=> debug50 ./compare
=> valgrind ./compare

=> size_of(int)
↳ function that returns size

Notes

Appointment




```
#include <stdio.h>
void swap(int *a, int* b);
int main(void)
{
  int x = 1;
  int y = 2;
  printf("x is %-i, y is %-i \n", x, y);
  swap(&x, &y);
  printf("x is %-i, y is %-i \n", x, y);
}
void swap(int *a, int *b)
{
  int tmp = *a;
  *a = *b;
  *b = tmp;
}
```

Notes

Appointment

13 June Friday

JUNE					2008
Mon	30	02	09	16	23
Tue		03	10	17	24
Wed		04	11	18	25
Thu		05	12	19	26
Fri		06	13	20	27
Sat		07	14	21	28
Sun	01	08	15	22	29

=> Heap overflow -

=> Stack overflow -

=> Danger of recursion - stack overflow

=> Buffer overflow - going beyond the array of allocated memory.

=> char s[4];

=>
```
#include <cs50.h>
#include <stdio.h>
#include <string.h>
int main(void)
{
    FILE *file = fopen("phonebook.csv", "a");
    if (file == NULL)
    {
        return 1;
    }
    char *name = get_string("Name:");
    chare *num = get_string("phone:");
    fprintf(file, "%s, %s\n", name, number);
```

"w" → write

"r" → to read

↓ to append

Notes

Appointment

```
    fclose(file);
}
```

=> jpeg.c

```
// Detects if a file is a jpeg
#include <stdint.h>
#include <stdio.h>
typedef ~~uns~~ uint8_t BYTE;

int main(int argc, char *argv[])
{
    // check usage
    if (argc != 2)
    {
        return 1;
    }
```

Sunday 15

```
    // open file
    FILE *file = fopen(argv[1], "r");
    if (!file)
    {
        return 1;
    }
    // Read first three bytes
    BYTE bytes[3];
    fread(bytes, sizeof(BYTE), 3, file);
```

Notes

Appointment

16 June Monday

JUNE 2008

Mon	30	02	09	16	23
Tue		03	10	17	24
Wed		04	11	18	25
Thu		05	12	19	26
Fri		06	13	20	27
Sat		07	14	21	28
Sun	01	08	15	22	29

```
// Read first three bytes
if (bytes[0] = 0xff && bytes[1] == 0xd8 && bytes[2] == 0xff)
{
  printf("Maybe \n");
}
else
{
  printf ("No \n");
}
}
```

=> cp.c

```
// copies a file.

#include <stdint.h>
#include <stdio.h>
# include <stdlib.h>
typedef uint8_t BYTE;

int main (int argc, char *argv[])
{
    // ensure proper usage
```

Notes

Appointment

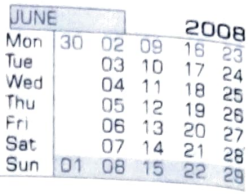
```
if (argc != 3)
{
  fprintf (stderr, "Usage: copy SOURCE DESTI
                                    NATION\n");
  return 1;
}
// open input file
FILE *source = fopen(argv[1], "r");
if(source == NULL)
{
  printf ("could not open %-s.\n", argv[1]);
  return 1;
}

// open output file
FILE *destination = fopen (argv[2], "w");
if(destination == NULL)
{
  fclose (source);
  printf ("could not create %-s.\n",argv[2]);
  return 1;
}
// copy source to destination, one BYTE at a time
BYTE buffer;
while ( fread(&buffer, sizeof(BYTE), 1, source))
      {
        fwrite(&buffer, sizeof(BYTE), 1, destination);
      }
```

Notes

Appointment

18 June Wednesday




```
// close files
fclose(source);
fclose(destination);
return 0;
}
```

=> uint8_t } datatypes → unsigned 8 Byte integers

=> int16_t } datatypes → signed 16 Byte integers.

=> fread(<buffer>, <size> <qty>, <file pointer>);

=> fwrite(<buffer>, <size>, <qty>, <file pointer>);

=> fopen

```
FILE * ptr = fopen (<filename>, <operation>);
FILE * ptr1 = fopen ("file1.txt", "r");
```

↓ read
w → write
a → append

Notes

Appointment

=> fclose

fclose (<file pointer>);
fclose(ptr1);

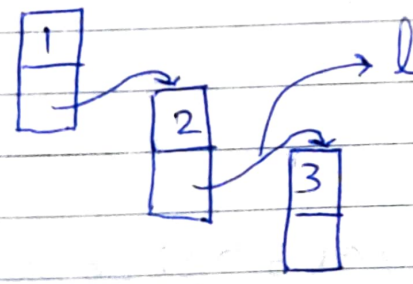
=> fgetc() - returns next character
- set operation to "r" to use this

char ch = fgetc(<file pointer>);

=> fputc() - append specific character.

fputc('c', <filename>);

=> Linked lists



-> typedef struct node
{
int number;
~~node *next~~
struct node *next;
}
node;

Notes

Appointment

→ node *n = malloc (size of (node));

→ if (n != NULL)
{
(*n). number = 1;
}

→ if (n != NULL)
{
n -> next = NULL;
}

→ node * n = malloc (size of (node));
if (n != NULL)
{
n -> number = 2;
n -> next = NULL;
}
list -> next = n;

⇒ int *list = malloc (3 * size of (int));
if (list == NULL)
{
return 1;
}
int *temp = realloc (list, 4 * size of (int));

↓

realloc automatically copies older elements from older array to new array.

Notes

Appointment