lets goto Silicone valley Date:

زبان C ← برنامه‌نویسی سطح پایین و زبان ماشین

برای نوشتن Hello world ← printf("helloworld!")

برای بیان یک توضیح ← /*.....*/ ← Comment

// ← توضیح استفاده

مثل stdioh

preproccsesor → راهنمای پیش‌پردازنده

include → اعلام به کامپایلر که در این برنامه از توابع کتابخانه‌ای استفاده شده است

نام یک فایل کتابخانه‌ای

میان دو علامت < >

Diamond

main ← نام برنامه

int main → تابع ورودی → ( ) ← main بعد از

نشان دهنده‌ی تابع بدون

مترجم

compiler
Interpreter

زبان سطح بالا
Basic و C

زبان ماشین

حداکثر طول شناسه‌ها ← ۳۱ و نباید کمتر از یک باشد

character → char

Subject: Date:

for integers → short, int, long

(اعداد صحیح)

for floating point → float, double, long double

(اعشاری)

* برای وارد کردن مقادیر از Scan f استفاده می کنیم.

Print f → داده های ورودی

عیناً چاپ می شوند مگر در مواردی که % دارند.

% → متناظر با هر متغیر

% & → چند کاراکتر

% d → عدد صحیح در مبنای ده

% u → عدد صحیح بدون علامت در مبنای ده

% f → اعداد اعشاری

% c → عدد کاراکتری

% % → symbole

% → mod on → در تقسیم باقی ماند

ادن هایی که در دلیوند قرار دارند → مال کتابخونه اس

ادن هایی که بین فوقتون رشته شده باشه → "

#include

نهال NAHAL

Subject: Date:

\n → new line

* در کلماتی که می‌خواهیم تعریف کنیم نباید با عدد شروع کنیم
* نباید @ داشته باشیم
* نباید فاصله بیاندازیم.

C: intrinsic data types:

* char → (character) → ذخیره‌ی کاراکترها (حافظه)
* int (integer) → برای استفاده‌ی اعداد صحیح
* float → 4 bytes (اعشاری هستند)
* double → 8 bytes

نکته: متغیرها نباید کلیدی باشند و نباید اولش عدد باشه.
در پایان تمام دستورات «;»

① میایم نوشتن رو مشخص می‌کنیم → ② معرفی و تعریف می‌کنیم → ③ مشخص کردن مقداری که داریم

تو char ← c = 'A'

نهال NAHAL

دستوراتی که قبل از توابع main نوشته می‌شوند

← دستورات پیش پردازنده ← و ندارند.

define → یک ثابت است که ← یا این که بعد از main

تغییر نمی‌کند

↓

متغیر + نوع + const

e++ → یک واحد اضافه

e-- → یک واحد کم شبه

باقی‌مانده = n عدد

می‌دهد.

برای باقی‌مانده استفاده می‌شود

* e==n → یعنی آیا e برابر با n هست یا نه

* e!=n → یعنی آیا e نابرابر n هست یا نه

* y=y+e $\xrightarrow{\text{برابر است با}}$ y+=e → انتساب جمع

y%=e → y=y%e → در جهت عمل اصلی صدق می‌کند

* y = e>5 ? 2 : 3 → اگر e>5 باشد y=2 اگر نباشد y=3

حالت شرطی

Subject: Date:

فایل های سرآیند همگی h دارند ← include

بین دو کوتیشن ← stdio.h فایل ← printf

char → c float → f اعداد صحیح → d

عدد را باید آدرس بدهیم ← scanf(" " , &)

آدرس متغیر را باید بدهیم تا بتواند بخواند

یعنی: ← برابر است می دهد

نکته: دستوری به نام getch وجود دارد که در توابع کتابخانه ای conio.ch قرار دارد. ← که به این صورت استفاده می شود:

* x = getch

ویژگی ای که دارد کاربر کاراکتری را که وارد می کند نمایش داده نمی شود و نیازی به کلیک کردن enter برای ادامه ی برنامه ندارد.

خط دستوری دیگری هم داریم به اسم getche که مانند getch است اما تنها تفاوت آن این است که کاراکتر وارده را نشان می دهد.

* دستور بعدی getchar هست که در تابع studio.h قرار دارد ← کاراکتری که می دهیم را نشان می دهد و با کلیک Enter ادامه می دهد.

Subject: Date:

* دستور بعدی putch است که در تابع کتابخانه‌ای conio.h قرار دارد. ← putch(x);

کارش مثل printf است با این تفاوت که فقط کاراکترها را نشان می‌دهد.

* تابع بعدی putchar است که در تابع stdio.h قرار دارد و نحوه‌ی کارش مانند putch است.

نهال
NAHAL

Subject: Date:

در scanf به صورت زیر عمل می‌کنیم:

scanf ("%d %d", &x, &y);

* یک space می‌دهیم

انواع ارورها: خطای نحوی → syntax error:
(دستور را تشخیص نداده)

خطای نوع داده → typechecking errors
انتظار

خطای زمان اجرا → تقسیم بر صفر → Runtime errors

نکته: دامنه‌ی اعشاری double بیشتر از float است.

نکته: /t ← tab می‌دهد

نکته: تقسیم دو تا int بر هم تقسیم می‌شوند int می‌دهد

نکته: p+++ نداریم!

در مقایسه‌ها ← جواب درست ① جواب غلط صفر

Subject: Date:

دستورات کنترل روند برنامه
و ساختار حلقه‌های تکرار :

* ساختارهای تصمیم
  - if
  - if else
  - switch

if (عبارت منطقی)
دستور ;

Example: if (x > 10)
x++;

if (عبارت منطقی)
{
دستور 1
دستور 2
...
دستور n
}

Example

if (x < y)
{
x = x + y ;
printf("%d, %d", x, y);
}

if .....
elsif
.....
else if

Subject: Date:

حلقه‌ی for ← وقتی که تعداد دفعات تکرار حلقه برای ما مشخص شده باشد

* for (گام حرکت ; شرط حلقه ; مقدار اولیه اندیس) *

حلقه

avg = (float) sum/n

نکته:

float در اینجا متغیر sum از نوع فقط است.

نکته:

%3.1f → رقم صحیح بعد از اعشار

تعداد ارقام + یک رقم اعشار بیشتر

حلقه‌ی while ←

while (شرط حلقه)

حلقه‌ی do...while ← حداقل یکبار دستورات را اجرا می‌کند.

do { دستورات

} → break → از حلقه آن خارج می‌شد.

Subject: Date:

* continue → یک دور رد می‌کند و دوباره شرط را چک می‌کند

switch → در واقع مثل همان if else می‌ماند

اما بخش default هم دارد.

while(شرط)
{ دستورات }

* تا در while و do while ← مثل دستورات یکبار اجرا می‌شوند

یعنی اول اجرا می‌شود سپس شرط حلقه پایان حلقه بررسی می‌شود

do {

} while();

* && → به معنای و است (∧)

نهال NAHAL

Subject: Date:

اگر برای if

switch (عبارت)

{

case (مقدار) : دستورات

break ;

نقطه‌ی دستور داشته باشیم.

لزومی ندارد که

default :

: دستورات

گذاریم.

☆ ☆

تابع نویسی: نوع تابع نام تابع (لیست پارامترها)

آدرس تابع

نوع هم

int main ()

{

فراخوانی تابع

نام تابع (لیست پارامترها) ;

اسم

}

عنوان تابع نوع تابع نام تابع (لیست پارامترها)

هم نوع هم اسم متغیر

{ دستورات

بدنه‌ی تابع

}

← void ← هیچ چیزی را برنمی‌گرداند.

Subject: Date:

* دستور Switch ← برای فهمیدن بررسی های چندگانه براساس مقادیر مختلف یک عبارت به جای if های متداول

```
Switch ( عبارت ) {
    case 1 مقدار
            دستورات 1
            break;
    case 2 مقدار
            دستورات 2
            break;
    deafult:
            دستورات n
}
```

در switch فقط تساوی را می‌توان چک کرد

اگر break استفاده نکنیم دستورات case بعدی و تا آخر اجرا خواهند شد

در حلقه‌ی for مقداردهی اولیه و گام وقتی می‌توانند شامل چندین دستور باشند ← بین آن‌ها باید از , استفاده کنیم.

* تبدیل حلقه‌ی while به for :

for (; while شرط حلقه‌ی ;)

{

بدنه حلقه‌ی while

}

تبدیل حلقه‌ی while به for (روش دیگر).

* for (; while شرط حلقه‌ی ; بدنه‌ی حلقه‌ی while)

توابع: در زبان C حدود ۱۲ ← Standard library طرح.

library <math.h>

* توان → pow (base, exp) ⇒

پایه توان

Ex: value = pow((i+1), -n); → $(i+1)^{-n}$

* رادیکال → sqrt (value)

Ex: root = sqrt (x*x + y*y) →

$root = \sqrt{x^2+y^2}$

نهال NAHAL

* fabs(value) ← قدر مطلق است!

* floor(value) ← جزء صحیح عدد اعشاری می‌کند.

→ header file

برخی از library ها عبارتند از: می‌گویند.

stdio.h , stdlib.h , math.h , time.h

↓ time library

character handeling → ctype.h

string library → string.h

local tools library → tool s.h

مثال اینهاست:

Return Type Function name (argument type parameter name);

اشاره گر ← pointer

int *p

→ int → *pi

p = &i

(*) p → محتوای آدرسی که p در خود ذخیره کرده است

محتوای عامل

(&) i → آدرس را مشخص کند

عامل آن نام یک متغیر

عامل & هر متغیر توسط سیستم عامل تعیین می شود

call by Reference

call by value

int f (int a , int b)

{ a++ ;
b++ ; }

main int a=۲, b = ۳ ;
f(a, b)
printf ("%d %d") → تاثیری بر روی a ندارد و همان ۲ را چاپ می کند

Subject: Date:

تابع Swap ← جابه‌جایی ۲ داده

بهتر است در توابعی که می‌نویسیم از IO استفاده نکنیم

scanf, printf f.

در string تا حداکثر ۳۱ = ۲ را حساب می‌کند.

getche چه تفاوتی با getchar

Subject: Date:

توابع یا functions ها:

برنامه نویسی پیمانه ای ← Modular Programming ← استراتژی تقسیم و غلبه

↓

Divide & conquer.

{ → function prototype

Int main

stdio.h → standard input & output

stdlib.h → توابع استاندارد

string.h → کاربار رشته ها

time.h → توابع مربوط به زمان و تاریخ

stddef.h

math.h → توابع ریاضی

locale.h → تنظیمات محلی

limits.h → محدودیت اعداد صحیح

floats.h → محدودیت اعداد اعشاری

ctype.h → توابع مربوط به کاراکتر

نهال NAHAL

Subject: Date:

assert.h → برای debug

errno.h → برای گزارش خطاها و خطایابی

لغت‌های خفن:

$$\lim_{n \to \infty} \frac{F_n}{F_{n-1}} = \frac{1+\sqrt{5}}{2} = \phi \simeq 1.618$$

در سری فیبوناچی

عدد طلایی!

* تابع fibonacci خیلی خفن اما:

```
int Fibo (int n)
{
    if (n==0) return 0;
    if (n==1) return 1;
    return Fibo(n-1) + Fibo(n-2);
}
```

تابع بازگشتی

Recersive function

تابع های سالم interation base ← سریع تر عمل می‌کند

و پیاده سازی سخت‌تر و مصرف منابع کمتر.

نهال NAHAL

متغیرها انواع اند

Global ← بیرون از همه توابع

local ← درون بدنه‌ی main و یا

Principle of least privilege ← اصل

اصل ارائه دادن کمترین امتیازات در توابع

تا وقتی که لازم نباشه Global نیستیم برای همه

auto register → automatic duration

auto ← default

register ← سرعت بالاست

یعنی بیاد اینکه کامپیلر غرق شد از بین می‌ره

extern , static → اینطوری بیشتر :)

extern → Global

static → حالت local دارد

static duration

هستن

تا وقتی برنامه اجرا شده در حافظه هستن :)

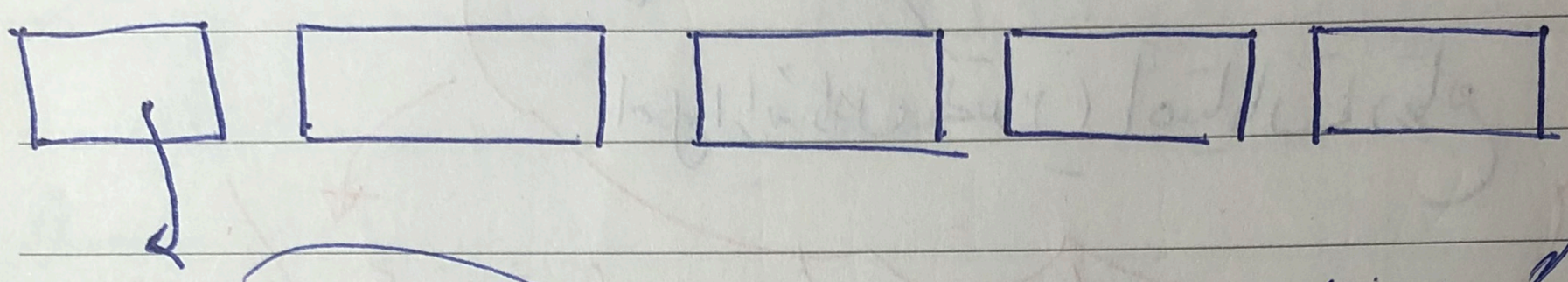
→ توان ۲ → 253 است.

Subject: Date:

* Arrays → تعدادی از المان‌ها هستند که از نظر نوع و اسم یکسان اند.

↓ یکی از امتدادهای داده به صورت static

int a[5];



a[0] → اندیس

نام آرایه

عددهای اندیس آرایه

* اگر چیزی نداریم → صفر است

int a[ ] → {۱, ۲, ۳, ۴} → 4

تا می‌شه

↳ safe

تراست!

* نامگذاری کردن → #define N 10

#include <stdio.h>

size of → byte داده را برحسب محاسبه می‌کند

برای پیدا کردن تعداد المان‌های آرایه → size of p / size of p[0]

نهال NAHAL

Subject: Date:

به علم اکوستیک شماره ی ۶ مراجعه شود. → برای اطلاعات بیشتر

* char s[] = "Hello", → shortcut یک نوع آرایه ای هست.

* char s[] = {'H', 'e', 'l', 'l', 'o', '\0'}

کاراکتر صفر یا NULL

String terminator

در string هایی که آرایه هستند نیازی به & نداریم اسم آرایه به جور خودش آدرس کسوبای نه.

خودش از جنس آدرس است.

Example: char name[10];
scanf("%s", ~~&~~ name);

جلوگیری از تغییرات ناخواسته → const

least privilege principle → آرایه ها را همواره به صورت const به توابع ارسال کنید مگر اینکه نیاز مند تغییر باشند.

الگوریتم Bubble sort ، ← از نوع n^2

* بهینه سازی شده ← $n.\log n$

* # دستوری داریم به نام printArray(, N) است.

نام آرایه ، تعداد

void Bubblesort (int a[], int N)

یکبار Sort می‌کند در هر بار

```
int swapped;
int i, j;
for (i = 0; i < N; i++)
{ swapped = 0;
    for ( j = 0; j < N-1; j++)
    {
        if (a[j] > a[j+1])
        {   temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
            swapped = 1;
        }
    } if (swapped)
        break;
}
```

آرایه

a[ ][ ]

سطر ستون

* C99 ⟶ VLA بشیابی ی‌کند از

variable length Arrey

* int a[N] ⟶ یعنی N می‌تواند متغیر باشد.

Pointers: aptr یک آدرس در یک متغیر از نوع int است.

int *aPtr

aPtr آدرس محل ذخیره سازی یک متغیر از نوع int است.

(۴) (۱)

(۳) (۲)

&a = aPtr

Address ⟶ operator عملگر

a = *aPtr

Dereferencing operator

آدرس‌ها را خنثی می‌کند

&*aptr = aptr

*&a = a

Subject: Date:

pointer ⇒ call by Refrence

* بسیار کاربردی هستند. *

* به کاربردن سیوتاکس Ptr در نام اشاره‌گرها

int *aptr

اشاره‌گر

از مقداردهی خالی می‌خواد

NULL و stdio.h
stddef.h *

printf(

آدرس را باید ← %p می‌شود.

	اشاره‌گر ثابت	اشاره‌گر متغیر
متوانایت	۱	۲
متوانستیر	۳	۴

int *aPtr → اشاره‌گر متغیر + متوانستیر

const int *aPtr → اشاره‌گر متغیر + متوانایت

int *const aPtr → اشاره‌گر ثابت + متوانای متغیر

آرایه عادی

نهال NAHAL

const int *const aptr → متغیر + اشاره گر ثابت

const آرایه

int a[ ];
int *const a;

→ معادل اند

a ≡ &a[0] *a ≡ a[0]

a+i ≡ &a[i]

*(a+i) = a[i]

استفاده از اشاره گرها (به ویژه در کار با داده های حجیم) سرعت اجرای برنامه ها را افزایش می دهد.

int a[ ]; → آرایه

int *aPtr; → اشاره‌گر

دقت بسیار مهم:

* a ≡ aPtr ≡ & a[0]

subscript Notation

a[i] ≡ aPtr[i] ≡ *(a+i) ≡ *(aPtr+i)

& a[i] = & aptr[i] = a+i = aptr+i

offset Notation

وقتی به یکی از خانه‌های آرایه اشاره‌ی کند می‌تواند به بقیه هم دسترسی داشته باشد.

unsign → بدون علامت است.

== → هم مقدار را چک می‌کند و هم نوع را

از مقدار رم استفاده می‌کند → define
و بجای تا
int وارد شده آن عدد را قرار می‌دهد.

INT_MIN → Library <limits.h>
کوچکترین عدد ممکن.

fflush (stdin) → برای اینکه بیند

رشته: str[] = mark X

str[] = {'m', 'a', 'r', 'k', \0}

gets → space
scanf → space هم می‌گیرد
space نمی‌گیرد

Strlen(a)

کار با رشته ها:

\0 → NULL character

Const char *s = char s[]

char s[20]

*scanf("%s", s); //unsafe

*scanf("%14s", s); //safe

<ctype.h> → کار با کاراکترها

<stdlib.h>

<stdio.h>

<string.h> → تابع رشته ای

Subject:

<ctype.h>

توابع آن → f(x) → character

→ EOF = -1

توابع تشخیص نوع کاراکتر

- isdigit → آزمایش عددی بودن
- isalpha → حرف بودن
- isalnum → عدد یا حرف بودن
- isxdigit → آیا کاراکتر وارد شده هگزادسیمال هست یا نه
  0, 1, ..., 9, a, b, ..., f
  A, b, ..., F
- islower → حرف کوچک
- isupper → حرف بزرگ
- is space → کاراکتر فاصله‌ای
- \r → carriage return
  برگرده اول خط
- vertical Tab → \v
- form Feed → \f
- iscntrl → کاراکتر کنترلی
- ispunct → جزء علائم نقطه‌گذاری
- isprint → عدد/حرف/فاصله
- isgraph → عدد/حرف/علائم
  فاصله ندارد

آنچه که isprint نباشد کاراکتر کنترلی است.

Subject: Date:

* if → به صورت جالب دیگر:

* شرط ? A : B ;

اگر درست → A

اگر غلط بود → B

condition? true statment: false statment;

توابع تبدیل رشته ها به مقادیر عددی:

double x = atof(s) → array to floating point

↳ 1.25

↓

#inc <stdlib.h>

توابع →
- atof → double
- atoi → int
- atol → long int

→ برای کل عدد صحیح کاربرد

Subject: Date:

char s[] = "1.2 -3.5 10.8";

strtof (char *s, char **p)

خروجی تابع است

strtod → double
strtof ↗

{ strtol → long
{ strtoul → unsigned long

stdio.h → getchar → ورودی کاراکتر
putchar → خروجی "

{ gets / fgets
{ puts / fputs

{ sprintf → به جای این که نتیجه رو چاپ کند در یک رشته ذخیره می کند
{ sscanf → به جای اینکه از ورودی دریافت کند از یک رشته دریافت می کند

* strlen → برای محاسبه ی طول رشته

Subject: Date:

*strerror → چاپ ارور و خطا

*

```
gets → char *gets(char *str);
```

fgets → char *fgets(char *str, int num,

File *stream),

↓

stdin می‌نویسیم به جای

بلد ایم.

با تابع می‌شه ارایه برگردوند فقط فرمت باید پیش‌فرض باشه

```
puts(const char *str);
```

```
sprintf(char *str, const char* format,..)
```

Subject: Date:

<string.h>

copy & cat → کپی و الصاق

↳ strcpy , strncpy ; strcat ; strncat

محدودیت در تعداد کاراکترها

الحاق دو رشته به هم

comparison → strcmp , strncmp

search → strchr ; strspn ; strcspn

دنبال کاراکتر

دنبال یک رشته

عدم وقوع رشته

strpbrk → مجموعه‌ای از کاراکترها و رشته‌ها و تقسیم رویه‌ای کنه

Subject: Date:

*

strcpy (char* destination, const char* source);

strncpy (char* destination, const char* source, sizeof);

strcat (char* destination, const char* source);

strcmp (const char* str1, const char* str2);

اولی بزرگتر ← جواب بزرگتر از صفر

دومی بزرگتر ← جواب کوچکتر از صفر

strchr (const char* str, int character);

Subject: Date:

قالب بندی های ورودی و خروجی :

*Integers: → types

d or i → decimal integers

بدون علامت ← u → unsinged decimal integer

عدد در مبنای هشت ← O → octal integer

X → hexadecimal

هشت

(مبنای ۱۶)

0, ..., 9, a, b, c, d, e, f

0, ..., 9 . AB, C, ...

*Integer modifiers:

h → short

l → long

*floating point numbers:

e or E → exponential 1.4e5

* f (or F) → fixed point 23.45

غایی

* g or G → e or f

براساس بزرگی عدد، خودش انتخاب می کند

Subject: Date:

→ l or L ; Length Modifier

Strings & Characters :

* c → char
* s → string (char*)

pointers

* p → pointer

#برای جا باز کردن عدد ٪ و یا رسوندن مفهوم آن از ٪٪ استفاده می کنیم.

٪ 4 d
→ width

٪ .2 f
→ precision → میزان دقت

Subject: Date:

نکته‌ی بسیار جالب توجه : *

*printf("%10.4f", 123.456789); *

printf("%.*f", 10, 4, 123.456789); *

- (minus sign) → left justification

+ (positive sign) → Displays + for positive number

" " (space character) → Displays a ␣ befor positive numbers

0 (zero) → padding zeros

1.5 000

صفرهای من

→ وقتی در مبنای هشت یا octal هستیم → 0: Adds "0" prefix

0 اضافه می‌کند

→ x or X : 0x or 0X

\a → Alert #

\b → Back space

\f → form feed

\r → به ابتدای خط می‌رود → سر خط

نهال NAHAL

Subject: Date:

Scan Set :

نکته‌ی خیلی جالب!

وقتی می‌خواهیم یه رشته از کاربر بگیریم ولی فقط می‌خواهیم از حروف خاصی باشه:

Ex:
```
char str[10];
scanf("%[aeiou]", str);
```

برنامه با همین ترتیب تنها قبول می‌کنه

Inverted Scan Set,

```
scanf("%[^aeiou]", str);
```

این عبارت روی فونه و وقتی به یکی از حاهای ریشه دیگه می‌خوره!

برای جدا کردن اعداد گرفته شده می‌توان از * استفاده کرد.

Subject: Date:

structures & Union :

شباهت میان ساختار و آرایه : جمع آوری داده ها تحت یک نام

تفاوت : آرایه ← مبتنی بر اندیس و شامل داده های هم نوع

ساختار ← مبتنی بر نام و شامل داده های نیز هم نوع

بعد از ۲ ← struct باید ؟ بنویسیم

```
struct student * ptsu1;
```

```
(*ptsu1).id or ptsu1->id
```

دسترسی از طریق اشاره گر

Arrow operator

دسترسی مستقیم

نقطه ← اولویت بیشتری نسبت به * دارد.

می توانیم struct توی struct داشته باشیم.

ساختار فرد ارجاع :

```
struct student
{struct student *pFriend;}
```

Subject: Date:

دستور typedef :

در واقع می‌توانیم اسم نوع‌ها را با این روش عوض کنیم:

```
typedef int Integer;
```

```
typedef struct studen Student;
```

به جای این ↑ می‌توانیم از این به بعد این بنویسیم

از طرفی این حالت هم شد:

```
typedef struct stud
{
}Student;
```

نکته: به طور پیش‌فرض func(s)

ساختارها به صورت ارزشی به توابع ارسال می‌شوند

برای فراخوانی به صورت مرجعی از اشاره‌گر به ساختار استفاده می‌شود

Subject: Date:

ساختارهای متحد Unions :

مثل structure می‌مونن.

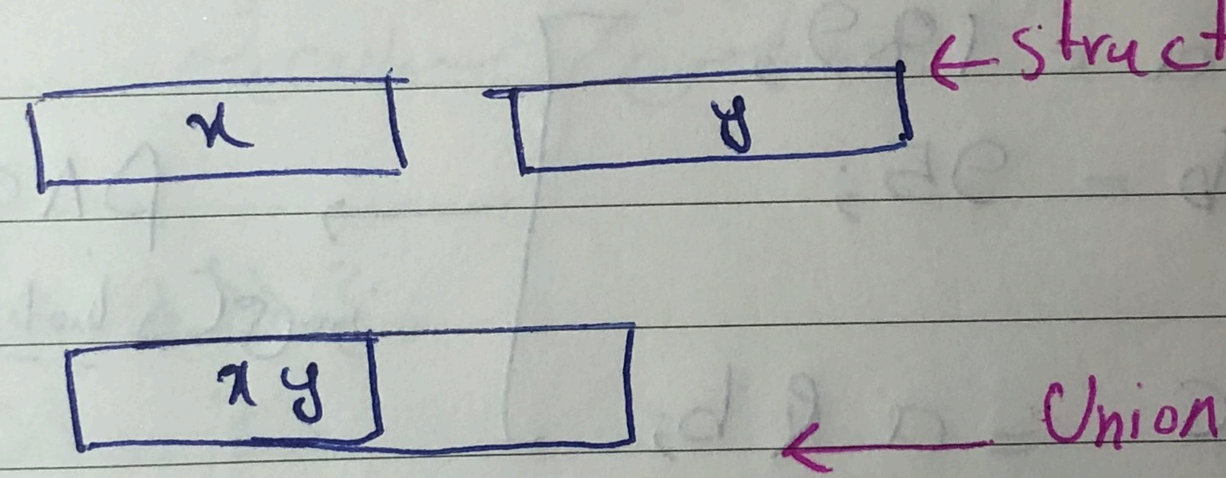
```
Union num
{
    int x;
    float y;
};
```

تفاوتش با structure اینه

که structure x و y رو کل زمیشون

جدا دارن ولی تو Union کل اشتراکی

دارند.



داده‌های شناور استفاده می‌شود یعنی ممکن است

که عدد int یا float باشد.

می‌توانیم در struct از Union استفاده کنیم.

Subject: Date:

Bit wise operators : علامت‌های بیتی

& Bitwis AND → &

| Bitwis OR → |

^ → Bit wise XOR → موقعی که باهم برابر باشند ۰ و موقعی که برابر نباشند ۱

~ → Bit wis ones complement

برعکس می‌کند

مثال

```
int a = 189;
int b = 95;
int c = a & b;
```

→ مثل p∧q و منطق ریاضیات می‌مون

```
  10111101
  01011111
  ---------
  00011101
```

→ عدد ۲۹ می‌شه!

مثل ~ → int c = ~a

```
  10111101
  01000010
```

a &= 10 $\xrightarrow{\text{برابر با}}$ a = a & 10

<< Left shift

>> Right shift

Example: int a = 189 0001011111 01

int b = a << 3; 1011110 1000

۳ تا می‌یارم جیپ به جا شون صفری اضافه م

سمت راستش هم استفاده می‌شه.

معادل

a = a >> 3;

a >> = 3;

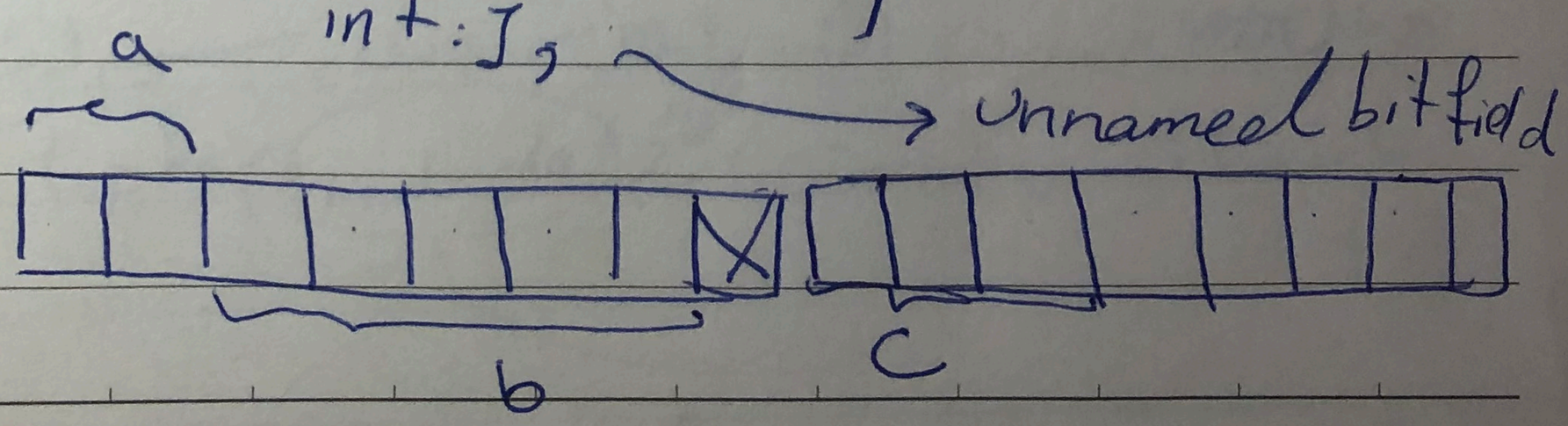
Bit fields: فیلدهای بیتی:

```
struct data
{ int a : 2;
  int b : 5;
  int c : 3;
  int : 1;
}
```

اندازه برحسب بیت

Bit field

unnamed bit field



Subject:

Date:

int : 0 ; → zero sized unnamed bitfield

به صورت پیش فرض
می‌رود بلوک بعدی

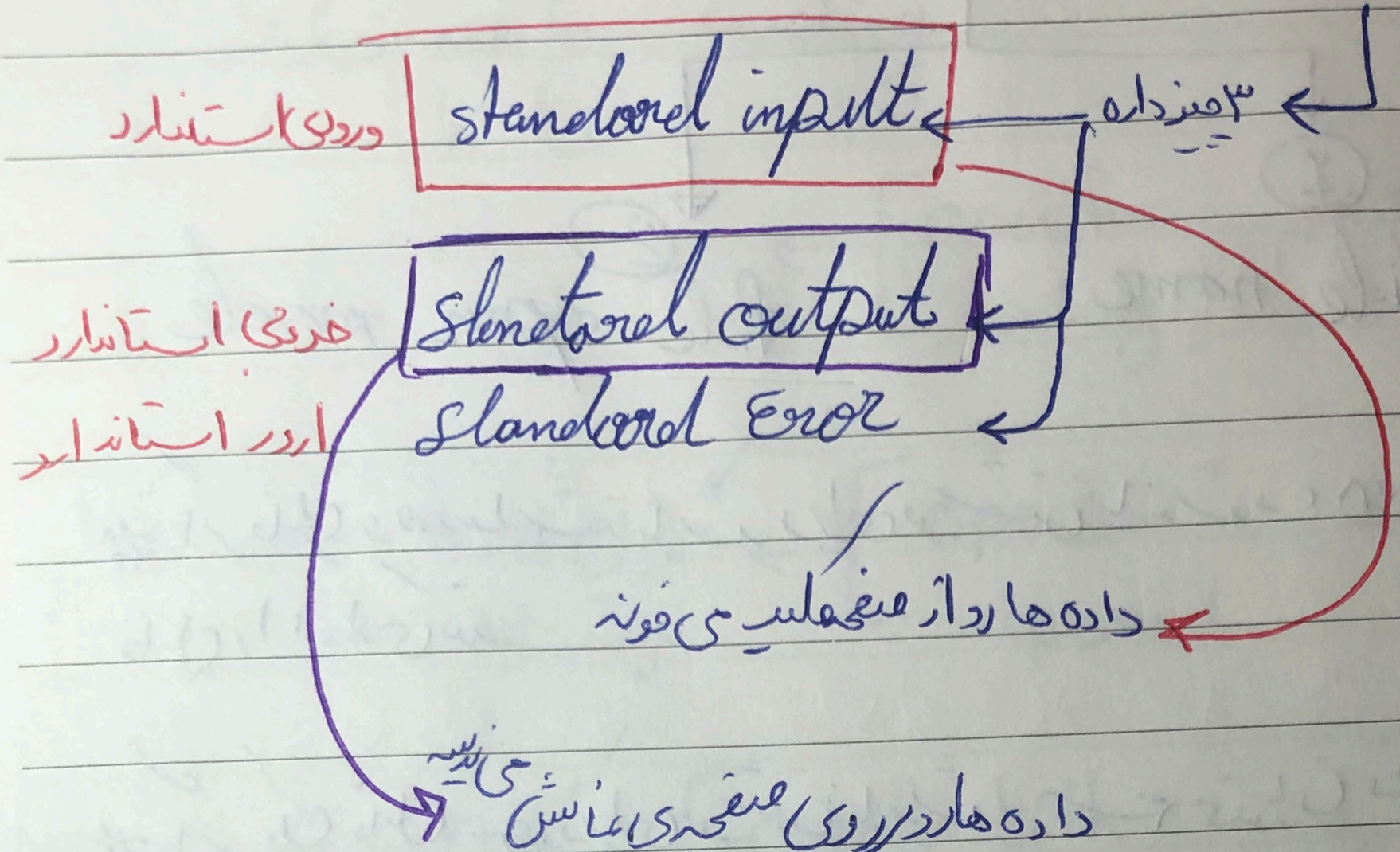
Enumerations:

داده های شمارشی:

Subject: Date:

Files:

فایل‌ها:



fgetc:

مانند getchar عمل کرده

یک اشاره‌گر FILE را برای فایلی که از آن کاراکتر خوانده می‌شود و به عنوان ورودی دریافت می‌کند

fputc:

هم مثل putchar یک کاراکتر را در فایل می‌نویسد

FILE *cfptr ← یک اشاره‌گر به ساختار فایل

هر فایل باز باید یک اشاره‌گر اعلان شده به عنوان مستقیم از نوع FILE داشته باشد ← برای دستری

fopen("clients.dat") ← نام

Subject: Date:

fopen → دو آرگومان را دریافت می‌کند

① file name

② file open mode

اگر فایلی وجود نداشته باشد برای نوشتن باز شود، fopen فایل را ایجاد می‌کند

اگر یک فایل موجود برای نوشتن باز شود محتویات قبلی‌اش پاک می‌شه

برای باز کردن فایل ← "w" استفاده می‌شود

fprintf و fputc با استفاده از stdout به عنوان آرگومان داده‌های خروجی را ارسال می‌کند.

fprintf(stdout, "%d %s %f, account, name, balance)

Subject: Date:

r* ← به معنای ← [Open for reading] ← اگر فایل وجود نداشته باشد Null برمی‌گرداند

w* ← [open for writing] ←

a ← [open for append] ← اگر فایل وجود نداشته باشد آن را می‌سازد.

r+ ← [open for both reading & writing] ← اگر فایل وجود نداشته باشد Null برمی‌گرداند.

w+ ← [open for both reading & writing] ←

a+ ← [open for both reading & appending] ← اگر فایل وجود نداشته باشد آن را می‌سازد.

fclose ← بستن فایل با نام

fscanf → ۳ تا ورودی داره

① → اشاره‌گر به فایل

② → فرمت داده

③ → آدرس متغیر

نهال NAHAL

Subject: Date:

rb ← یک فایل موجود برای خواندن در دودویی

wb ← یک فایل را برای نوشتن در حالت دودویی

ab ← اطلاعاتی را به انتهای فایل اضافه می‌کند یک فایل را برای نوشتن در انتهای فایل در حالت دودویی باز یا ایجاد می‌کند.

rb+ ← یک فایل موجود را به روزرسانی (خواندن و نوشتن) در حالت دودویی باز می‌کند.

Subject: Date:

#get w → Reads an integer from a file;

Put w → writing an integer to a file

fseek →

ftell → Returns

rewind → sets the file pointer at the begeing of file.

fputc (char file, file_pointer)

↳ writes a character to the file

fputs (str, file_pointer)

↳ writes string to the file

Subject: Date:

malloc → تخصيص ديناميكي :

```
newPtr = malloc (sizeof(structnode));

free (newPtr);
```