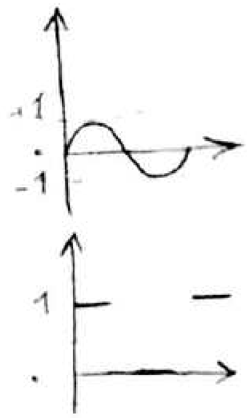
computer

Subject: مبانی **Date:** ۷/۱۵

Tavakoli.Tn@gmail.com

هر جلسه امتحان 😐
پنج شنبه ۳/۵ - ۱ هفته بعد
مبانی اولیه کامپیوتر مطالعه شود!



سیستم ← آنالوگ
سیستم ← دیجیتال

(یا صفر یا یک / یا روز یا شب / یا روشن یا خاموش)

		حالت درست
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5		0101
6		0110
7		0111
8		1000
9		1001

کوچکترین واحد حافظه در کامپیوتر bit (تک، ذره)

* در هر مبنا بزرگترین رقم یکی کمتر از آن مبنا باشد.

واحدهای حافظه

bit	۲
byte	8
KB (۱۰۲۴)	1024
MB	1024
GB	″
TB	″
PB پتابایت	
EB اگزابایت	

مثال: یک کامپیوتر دارای حافظه‌های 2 TB و 512 mB و 256 GB می باشد حافظه‌ی این pc چند TB و KB تشکیل شده است؟

512mB   256GB   2T

$\frac{512}{2^{20}} + \frac{256}{2^{10}} + n$

(Bench mark : محک و معیار)

Subject. Date:

0 یعنی خاموش
1 یعنی روشن

Set : 0 → 1 (تبدیل 0 به 1)
Reset : 1 → 0 (تبدیل 1 به 0)

انواع حافظه:
- حافظه اصلی : حافظه ای که مستقیم با CPU در ارتباط است
  Ram / Rom
- حافظه جانبی : felash, Hard

تفاوت الکتریکی و الکترونیکی ؟

RAM : حافظه ای با دسترسی به صورت تصادفی (سریع تر)

خصوصیات ← با قطع برق اطلاعات از بین میره
← فوق العاده سریع

RoM : (بالا آوردن Boot سیستم) / (برنامه ریزی آن دست ما نیست)
← حافظه ی فقط خواندنی
← با رفتن برق اطلاعات حفظ می شود

نوع خاص ROM ← PROM : حافظه ی فقط خواندنی که قابلیت برنامه ریزی دارد
(یعنی یک ROM خوام کردم)

← (سر سیم ها فیوز قرار می دهند که فیوز ها هم سالم هستند و برنامه ریزی می شوند یعنی فیوز ها که می خوایم رو می سوزونیم تا جریان فقط در سیم هایی که می خوایم وجود داشته باشه)

EP RoM : حافظه ی فقط خواندنی که قابلیت برنامه ریزی مجدد نیز دارد

← (فیوز ها رو نمی سوزونیم برای برنامه ریزی بلکه موقتا با اشعه ی ماورای بنفش قطع می کنیم)

EEPROM : همون EPROM می باشد که به صورت ~~[crossed out]~~ الکتریکی Reset می شود (مانند دکمه ی reset پشت مودم)
دیگر نیازی به اشعه از ماوراء بنفش نیست

زبان های برنامه نویسی ← سطح بالا : زبان هایی هستند که به زبان محاوره ای انسان نزدیک می باشند مثال؟ VB و C sharp
آسان تر (هر خط دستور همان شامل چندین خط دستور است)
اگر دلیلی برای max بهره وری از cpu نباشد.

سطح پایین : زبانی که به زبان ماشین (Pc) نزدیک می باشد مثال؟ اسمبلی
سخت تر (هر خط دستور تنها و تنها شامل یک خط دستور است)
برای max استفاده (بهره وری) از cpu از زبان سطح پایین استفاده می شود

A	10
B	11
C	12
D	13
E	14
F	15

* مبنا (۲) و (۱۶) و (۱۰) در PC کاربرد دارد.
(۱۶ چرا؟) ← برای آسان تر شدن

D 13 → در یک واحد یک جا می توانیم بذاریم ۱۰ ← جاش می ذاریم A که راحت تر نیز هست.

* در سیستم های عددی موقعیت مکانی هر رقم دارای ارزش معین می باشد
با این تعریف می توانیم بگوییم که روش ارزش مکانی برای تمامی مبناها صادق است.

● ۸۴۲
۸ یکان ← دسته های یکی
۴ دهگان ← دسته ها ۱۰ تایی
۲ صدگان ← دسته های صدتایی

● $(346)_7$
۶ ← ۱ تایی
۴ ← ۷ تایی
۳ ← ۴۹ تایی

● 10011101
دسته های یکی، دوتای، چهارتایی، ۸تایی، ۱۶تایی ...

← مبنا بود : (۱۰)
← CBA راست : (۱۶)
← ۰ و ۱ بنا بود : (۲)

Subject: Dale:

* جهت تبدیل اعداد در هر مبنا به مبنای دیگر ابتدا عدد اولیه را به مبنای ۱۰ برده سپس از مبنای ۱۰ به مبنای مورد نظر می بریم.

مثال

$$(۳\ \ ۲\ \ ۱)_{۵} = (\ ؟\ )_{۸}$$

(۱) جهت تبدیل عدد از هر مبنا به مبنای ۱۰ با استفاده از الگوی زیر عمل می کنیم:

$$\sum (\text{ارقام} \times \text{مبنا}^{\text{توان}})$$

$$(۱ \times ۵^{۰}) + (۲ \times ۵^{۱}) + (۳ \times ۵^{۲})$$

$$۱ + ۱۰ + ۷۵ = ۸۶$$

(۲) جهت تبدیل عدد از مبنای ۱۰ به هر مبنای دیگر از روش تقسیم های متوالی استفاده می کنیم

در این روش عدد مورد نظر را تقسیم بر مبنای خواسته شده می کنیم سپس در هر مرحله خارج قسمت را مجدداً بر مبنا تقسیم می کنیم و این عمل را تا جایی ادامه می دهیم که خارج قسمت [کوچکتر از مبنای مورد نظر باشد (خارج قسمت قابل تقسیم صحیح نباشد)]

نوشته ی قدیم

← (۱) صفر سمت چپ سمت راست به چپ باقی مانده ها را می نویسیم

← (۲) قابل تقسیم صحیح نباشد سمت چپ سمت راست

به چپ آخرین خارج قسمت و باقی مانده ها را می نویسیم

$$(۱\ \ ۲\ \ ۶)_{۸}$$

Subject: Date:

مثال

$$\begin{array}{r} (۱۷۶۴۳ \\ +۳۲۷۷ \\ \hline ۱۳۱۴۲ \end{array})_۸ \qquad \begin{array}{r} (F۲۹۴۴ \\ -۹۹۹۹ \\ \hline 58AB \end{array})_{۱۶}$$

carry

Data → Process → Information این چیه؟

جلسه دوم

با ارزش ترین بیت (سمت چپی) ← بیت علامت یا

بیت علامت → + : 0 ، - : 1

تعداد بیت‌ها / ظرف‌ها ← 2^{n} تعداد حالت‌ها

به دلیل دودویی بودن PC (۰ و ۱)

متمم (جمع ۹۰)
مکمل (جمع ۱۸۰)
← حالا دیگه می‌گیم در PC چون فقط ۰ و ۱ داریم در هر ۲ صورت می‌شود ۰ و ۱

$$\begin{array}{r} 1010 \\ -1111 \\ \hline 0101 \end{array}$$
→ متمم ۱ ← تمام صفرها یک می‌شود تمام یک‌ها صفر

متمم 2 ← برای بدست آوردن متمم 2 ابتدا متمم ۱ را بدست آورده سپس با ۱ جمع می‌کنیم

$$\begin{array}{r} 1010 \\ -1111 \\ \hline 0101 \\ +\ \ \ 1 \\ \hline 0110 \end{array}$$
0101 → متمم ۱
0110 → متمم 2

* در PC ما فقط یک مدار جمع کننده داریم و بقیه اعمال ریاضی رو می‌سازیم

A − B → A > B ، B > A

$$\begin{array}{r} 0101 \\ -1001 \\ \hline 0001 \end{array} \qquad \begin{array}{r} 10 \\ -9 \\ \hline 1 \end{array}$$

MICRO

Subject. Date:

به منظور انجام عمل تفریق با استفاده از روش متمم ۲ اگر عدد اول بزرگتر از عدد دوم بود ابتدا متمم ۲ عدد دوم را به دست آورده سپس با عدد اول جمع می کنیم و سپس از رقم نقلی بوجود آمده چشم پوشی می کنیم و حاصل به دست آمده

A	1010	10	متمم عدد دوم → 1001
B	-1001	9	0110 → متمم اول
	0001	1	0111 → متمم دوم

1010 + 0111 = (1)0001

رقم نقلی — جواب اصلی

حال اگر عدد دوم بزرگتر از عدد اول بود ابتدا متمم ۲ عدد دوم را به دست آورده سپس با عدد اول جمع می کنیم سپس حاصل به دست آمده متمم ۲ می گیریم

	0100	4	
	-0111	-7	
	0011	3	

0111 → 1000 → متمم ۱ → 1001 → متمم ۲

0100 + 1001 = 1101

1101 → 0010 → متمم ۱ → 0011 → متمم ۲

* علامت عدد به دست آمده منفی می باشد

جواب اصلی

تبدیل مبنا به مبنای دیگر (اعشاری)

از مبنی دای خواهیم ببریم به ۲ (رقم صحیح را جدا تقسیم کرده و رقم اعشار را ابتدا جدا تا حاصل ضرب ۲ می گیریم به رقم اعشار کسر کنار هم می گذاریم)

از این مبنا

$(12.625)_{10} = (\quad)_2$

12 ÷ 2 = 6 باقی 0
6 ÷ 2 = 3 باقی 0
3 ÷ 2 = 1 باقی 1
1 ÷ 2 = 0 باقی 1

1100

0.625 × 2 = 1.25
0.25 × 2 = 0.5
0.5 × 2 = 1.0

1100.101

حال از این کلاس

$(10.011)_2 = (2.375)_{10}$

2^{-1} 2^{-2} 2^{-3}

$(1 \times 2^{-1}) + (1 \times 2^{-3})$

$(.125) + (.1125) = .375$

☺ رقم صحیح رو داریم ضرب می کنیم برای قسمت اعشار در تقریبی لیم ... سمت راست عدد را معمول عدد صحیح بوده حالا سمت چپ عدد را می نه عدد اعشاری که منفیه پس معمول راه قبلی رو برای تبدیل مبنای عدد ۱۰ انجام می دهیم با این تفاوت که توان منفی می باشد.

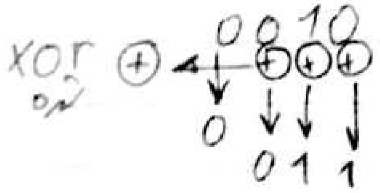
سیستم های مختلف کدی : همینگ

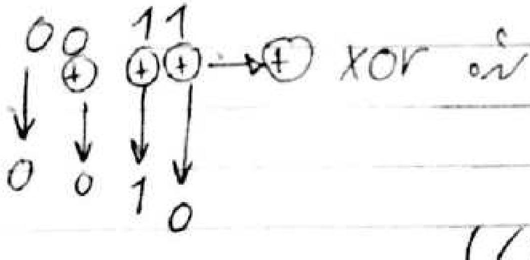
همینگ ↓ هم تشخیص، هم تصحیح

حروف کامل شود

Gray (گری) ← هر ۲ عدد متوالی در تعداد یک bit می توانند تفاوت داشته باشند)

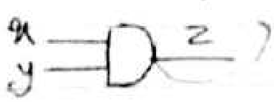
	binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101





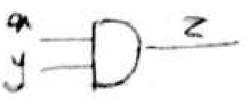
گیت ها (ورودی)

و (AND)



x	0	0	1	1
y	0	1	0	1
z	0	0	0	1

یا (OR)



x	0	0	1	1
y	0	1	0	1
z	0	1	1	1

ی OR انحصاری (XOR)

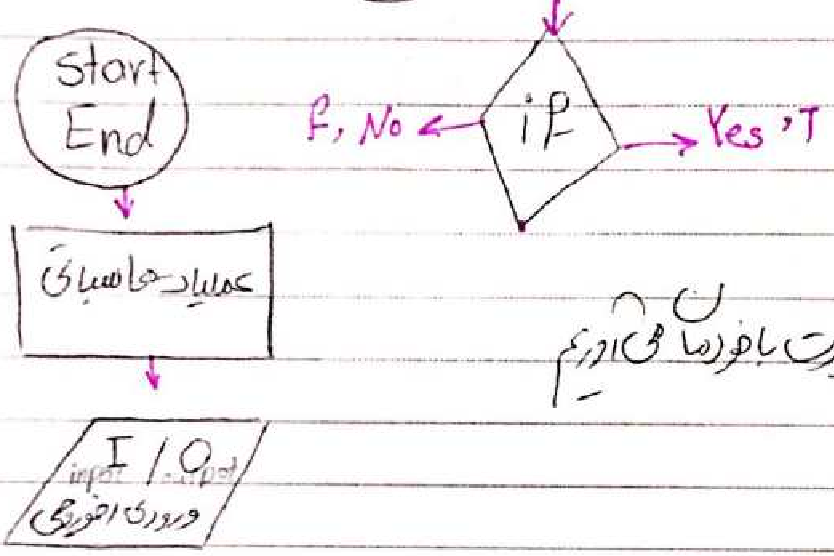
x	0	0	1	1
y	0	1	0	1
z	0	1	1	0

★ تابع XOR یک تابع فردی باشد یعنی اگر تعداد ورودی ها فرد باشد خروجی ۱ می شود در غیر اینصورت خروجی صفر می شود!

Subject. Date:

الگوریتم: روند اجرای یک سری عملیات به صورت خاص هم باید داشته باشد

برای مثال: الگوریتمی بنویسید که به یک ربات فرمان دهد تا با استفاده از قطعات چوب ۲m برای زمین مستطیلی به ابعاد ۲۵ در ۴۰ حصار بسازد. ارتفاع حصار ۱.۲ را فاصله بین ۲ حصار ۲m باشد.

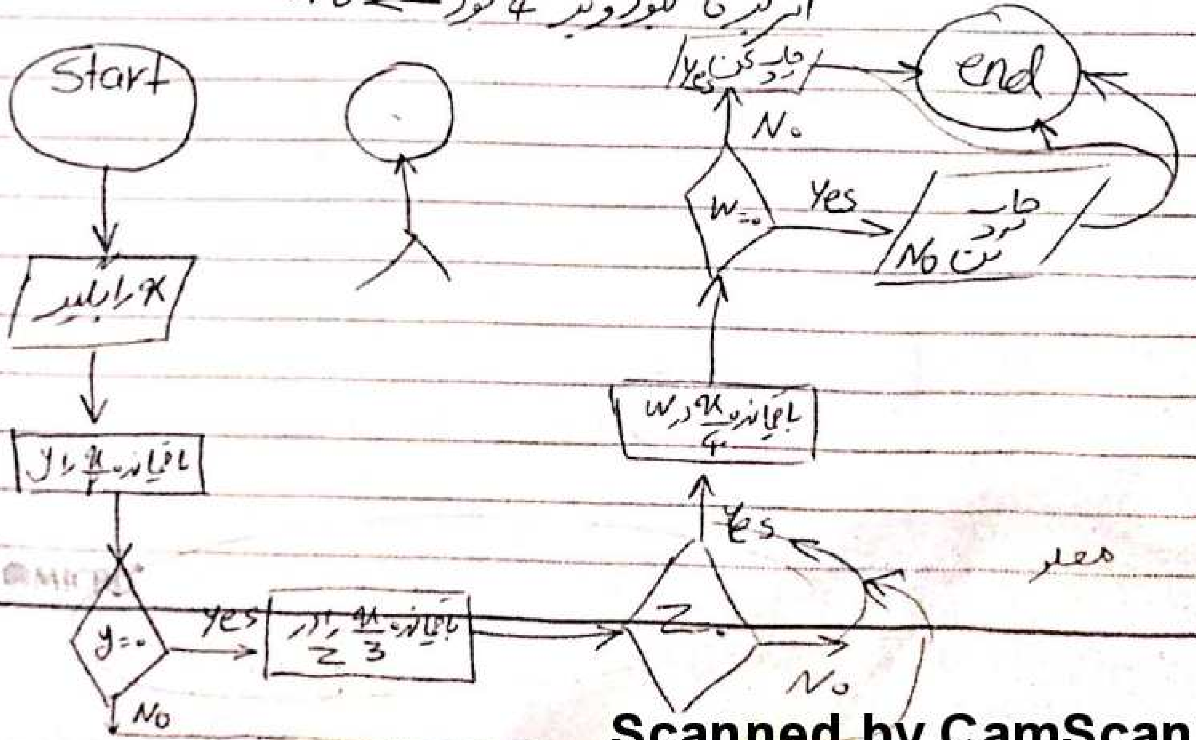


یک عدد الگوریتم و فلوچارت با فرمان می‌دهیم

شرطی (۱/۸/۹۵)

۲ ساختار داریم:
- if (اگر روند را این)
- if than else (اگر روند را این یعنی روند این)

ex: عددی را دریافت کند اگر بر 6 بخش پذیر بود و بر 4 نبود ← yes
اگر بر 6 نبود و بر 4 بود ← NO



Subject: Date:

به منظور حذف دستور go to جهت انجام دستورات برنامه‌ای ساخت یافته روی آوردیم
بدین منظور هر برنامه را بدون دستور برنامه‌ای تنها با استفاده از ۳ ساختار کلی انتخاب،
کنترلی و تکرار goto را پیاده می‌کنیم.

ساختار ترتیب ← اجرای دستورات به صورت متوالی
انتخاب ← به برنامه نویس اجازه می‌دهد تا براساس درستی یا نادرستی یک شرط تصمیم بگیرد
که دستورات اجرا شود
تکرار ← مجموعه‌ای از دستورات که تا زمانی که شرط خاصی برقرار است انجام می‌شود.

← به طور کلی ساختار C++ از ۳ بخش تشکیل شده:
1. محیط برنامه نویسی جهت نوشتن و ویرایش دستورات
2. compiler
3. توابع کتابخانه‌ای و دستورات

← مراحل انجام برنامه نویسی عبارتند از:
1. write
2. save
3. pre check ← بازبینی و حل کردن
4. compile ← ترجمه به زبان ماشین
5. Link ← پیوند به زبان ماشین
6. Load on RAM ←
7. Execute by CPU

خطاها ← ① خطاهای زمان ویرایش (نحوی)
② خطاهای منطقی یا زمان اجرا

① خطاهای زمان ویرایش:
خطاهایی که در اثر عدم رعایت دستورات زبان C++ یا خطا در type بوجود می‌آید
(در زمان compile مشخص می‌شود)

MICRO

(2) خطاهای زمان اجرا:

← (مثلا به برنامه نوشتم برای یک کلاسیت تا 99 نفر برنامه ok بوده شده error 100 داده یعنی برای اعداد 3 رقمی به مشکل خورده)

هرگاه در زمان اجرای برنامه خطاهایی که برای مثال در طراحی الگوریتمی وجود داشته باشد بوجود آید خطای منطقی رخ داده و این خطاها بدهستند

(خطاهای زمان اجرا) خطاها به دو دسته ی دیگر تبدیل می شوند

عملیات اجرایی

مهلک: باعث فاصله متوقف می شود و خطا را از این می دهد/خطای تقسیم بر صفر

غیر مهلک: خطایی که برای برنامه ادامه پیدا می کند ولی اجرا نتیجه اشتباه تولید می کند

نامگذاری: در نامگذاری می توانیم از حروف انگلیسی و اعداد و order line استفاده کنیم.

* یک نام می تواند از چند حرف بدون فاصله (space) تشکیل شده باشد.
* نام نمی تواند با عدد شروع شود
* در C++ حساسیت به حروف بزرگ و کوچک وجود دارد.
* در انتهای تمامی دستورات باید از ; (semicolon) استفاده کنیم
* طول هر نام نباید بیش از 32 کاراکتر باشد.
* نباید از کلامات کلیدی باشد (مثل if)
*

انواع داده اصلی :

① char یا کاراکتر ← جهت ذخیره داده‌های کاراکتری مثل / A, B, C, D
و بازه‌ی قابل قبول 127+ تا 128- را در بر می‌گیرد

② int (مخفف integer) ← ذخیره اعداد صحیح و بازه‌ی -32768 تا +32767 را در بر می‌گیرد.

③ float ← ذخیره اعداد اعشاری با دقت کم (4byte / 32 bit)

④ Double ← ذخیره اعداد اعشاری بزرگتر از float (8byte / 64 bit) long fact

اعمال ریاضی : + − = × ÷

اعمال انتساب (مقدار دهی، نسبت دهی)

a - = 5
a = a - 5

src → مبدا
dest → مقصد

مثال / Sum = Sum + 1

عملگرهای جمع و تفریق ← I++ (یک واحد اضافه کن)
I-- (یک واحد کم کن)

I++

I--

I++ ← اول اضافه بعد ها
++I ← اول ها بعد اضافه

MICRO

Subject:

Date:

عملگر % ← باقی مانده تقسیم صحیح دو عدد

10%2 → 0

10%4 مثال

a/=5 a = a/5

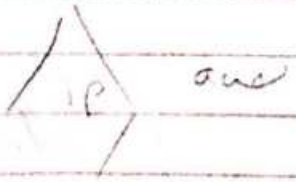
a/=5 /= عملگرهای پیش افزایشی ++I

پس افزایشی I++

عملگرها منطقی

x	y	and &&	or ||	!
0	0	F	F	T
0	1	F	T	T
1	0	F	T	F
1	1	T	F	F

x&&!y



Subject: Date: 95 .8.13

عملگرهای رابطه‌ای (>، <، =) ← با استفاده از این عملگرها می‌توانیم رابطه‌ای بین یک یا چند متغیر، با عبارت و یا ترکیبی از متغیرها و عبارات را برقرار کنیم.

* در محاسبات و عبارات باید همواره تقدم عملگرها را رعایت کرد.

عملگرهای رابطه‌ای عبارتند از :

==	برابری
!=	نابرابری
<	کوچکتر
>	بزرگتر
<=	کوچکتر مساوی
>=	بزرگتر مساوی

* توجه داشته باشید که عملگر = با == برابر نیست. اولی انتساب و دومی برابری.
هرگاه از = استفاده کنیم به معنای انتساب مقداری به متغیر مورد نظر می‌باشد؛ اما هرگاه از == استفاده کنیم منظور بررسی تساوی بین ۲ مقدار می‌باشد.

عملگرهای شرطی ← به منظور کنترل روند اجرای برنامه مورد استفاده قرار می‌گیرد.
بدین منظور می‌توانیم پس از ارزیابی عبارت شرط حالات ممکن (تعریف شده) را انجام دهیم.
فرم کلی به صورت : (عبارت شرط ? نتیجه 1 : نتیجه 2)

? جایگزینی دو حالت if

(a==5 ? 3 : 4) : Ex

← اگر a برابر 5 بود چاپ کن 3 در غیر اینصورت چاپ کن 4

این دستور معادل با ساختار if else می‌باشد؛ با استفاده از دستور if می‌توانیم روند اجرای برنامه را پس از بررسی شرط به سمت دستور یا مجموعه‌ای از دستورات مورد نظر هدایت کنیم. ساختار کلی دستور به صورت if (شرط) { } می‌باشد.

{ ← مجموعه‌ای از دستورات
}
else
{
}

Subject: Date:

دستگاه های ورودی و خروجی : موس، کیبورد، اسکنر، میکروفن
← صفحه نمایش، پرینتر

دستورات ورودی و خروجی : با استفاده از دستورات Cin و Cout می توانیم داده را از ورودی بگیریم و در خروجی نمایش دهیم؛ بدین منظور هر آنچه که پس از دستور Cin آورده شود (Cin >> a;) و در خروجی (Cout << a;) نمایش داده می شود.
این دستور هر آنچه در متغیر a باشد را چاپ می کند.

C in → ورودی
C out → خروجی

Ex : Cout << "Hi"; ← فقط Hi رو چاپ می کند 🙂

* در هنگام استفاده از Cout هر آنچه که داخل double cotation ("...") باشد عینا نمایش داده می شود.

Ex : برنامه ای بنویسید که نام و سن شما را در خروجی نمایش دهد.

```
# include <iostream.h>
# include <conio.h>
using namespace std;
int main ( )
{
   int a = 20 ;
   Cout << "Amir Tavakoli." ;
   Cout << "a" ;
   return 0 ;
}
```

توابع کتابخانه ای

int a ;
a = 20 ;
→ به جای تعریف می شد نوشت

تعریف متغیر با مقداردهی اولیه

هر چه داخل "" باشد عینا نوشته شده.

پایان

(int) a ; int a = 20
a = 20

MICRO

Subject: Date:

Ex: برنامه‌ای بنویسید که 3 عدد را از ورودی بگیرد و حاصل جمع، ضرب و میانگین را نمایش دهد.

```
# include <iostream.h>
# include <conio.h>
 using name space std;
  int main()
  {
  int a,b,c;
  Cout << "plz Enter 3 numbers";
  Cin >> a >> b >> c
  int sum ,mul;
  float avg;
 sum = a+b+c
 Cout << sum is= << sum << endl ;
 Cout << sum << endl
  mul =a*b*c;
  Cout << mul << endl;
  avg = (a+b+c)/3;
  Cout << avg << end l;
   return 0;
  }
```

Ex: برنامه‌ای بنویسید که طول و عرض دستمال را از ورودی بگیرد و محیط و مساحت را نمایش دهد.

```
# include <iostream.h>
# include <conio.h>
 using name space std;
  int main()
 {int a,b,p,s
 Cout << "plz Enter toolo Arz"; << endl
 Cin >> a,b;
 P = (a+b) * 2;
 cout << p is= mohit << endl;
```

MICRO

```
 S = a*b
 cout << s is= masahat << end
  return 0
```

Subject. / Dale:

Ex : برنامه‌ای بنویسید که شعاع دایره را دریافت کند؛ محیط و مساحت را محاسبه و نمایش دهد.

```
#include <iostream.h>
#include <conio.h>
  using name space std;
#Define p; 3.14          // Define
   int main ()
   {
    float r, s, Pr
    Cout << "enter r " << endl
    Cin >> r;
    S=P*r*r
    Pr=2*P*r
    cout << "masahat" << S << endl
    Cout << "mohit" << Pr << endl
     return 0;
   }
```

Ex : برنامه‌ای بنویسید که یک عدد اعشاری را از ورودی بگیرد و نزدیک‌ترین عدد صحیح به آن را نمایش دهد.

```
#include <iostream.h>
#include <conio.h>
using name space std;
 int main ()
{
 float a,f;
 Cout << " Enter number" << endl;
 Cin >> a;
 int b=a;
 f=a-b;
 if(f>.15);
  b++;
  Cout << b;
  else
 Cout b;
```

MICRO

EX : برنامه‌ای بنویسید که 3 عدد را از ورودی بگیرد و MAX را نمایش دهد.

```
# include <iostream.h>
# include <conio.h>
using name space std;
int main()
{int a,b,c
Cout << "Enter 3 numbers" << endl;
Cin >> a,b,c;
{ if (a>b)
  if (a>c)
  Cout << "max" << a << endl;
  els
    Cout << "max" << c << endl;
    if (b>c)
    cout << "max" << b << endl;
      return 0
}
```

* ساختارهای کنترلی به منظور کنترل و انتقال روند اجرای برنامه به فرمی رود مانند ساختار دستوری شرطی if else که شرط را بررسی می کند و در صورت برقراری و عدم برقراری شرط فرمهای مختلف را انجام می دهد.
* دستور goto بدون بررسی شرطی روند اجرای برنامه را منتقل می نماید.
* دستورات switch و case که در صورت برقراری شرط یکی از انتخاب های ممکن صورت می پذیرد و تا زمانی که به دستورات Break برنخیم عملیات ادامه دارد دستور Countinu باعث می شود دستور یا مجموعه دستورات همچنان ادامه پیدا کند.
* دستور for برای ایجاد حلقه جهت اجرای دستور به صورت تکراری به فرمی رود و تا زمانی که تعداد دفعات تکرار حلقه معلوم باشد به فرمی رود.
  ← حالا اگر تعداد دفعات معلوم نبود از Whiledo استفاده می کنیم.

MICRO

Subject.

Dale:

Ex : 3 عدد وارده max و min را غالبی دهد. (راه حل ساده تر مشکل قبل)

```
min = a
max = a
if (b > max)
{ max = b;
}
else
if (b < min)
{ min = b;
}
if (c > max)
{
```

استفاده از دستور for

برای n تا عدد از حلقه‌ی for استفاده می‌کنیم

for را بعد از معرفی max و min و اینا قبل از کلیا قرار می‌دهیم

```
for(i=0; i<100; i++)
{ Cin >> n;
  if(n > max)
  { max = n;
  }
  else
  {min = n;
  }
}

for(i=0; i<100; i++)
for(i=100; i<500; i--)
```

Subject: Date:

(Case, switch)

EX: قطعه برنامه ای بنویسید که یک عدد از ورودی بگیرد اگر ۰ تا ۶ بود معادل روزهای هفته را چاپ کند در غیر اینصورت پیغام خطا نمایش دهد

0 → شنبه

1 → یکشنبه

? → —

```
int k;
Cin >> k;
swich (k)
{ case 0: {cout << "shanbe; break;}
  case 1: {cout << "Yekshanbe; break;}
  :
  case 6: { ...
  default : cout "error!";
}
getch();
```

getch(); ← یه کاراکتر میگیره

default

★ در صورت وجود شرایط پیش بینی نشده می توانیم حالت دیفالت را در نظر بگیریم

EX: قطعه برنامه ای بنویسید که یک کاراکتر از ورودی بگیرد و ۴ رنگ دلخواه را در نظر بگیرد اگر کاراکتر حرف اول هر کدام از رنگ ها بود رنگ را چاپ کند.

Ex

Subject:

Dale: / / /

Ex : حقیقی برنامه ای بنویسید که یک عدد از ورودی دریافت کند اگر عدد 5 یا 9 بود چاپ کند ok در غیر اینصورت error.

Ex : حقیقی برنامه ای بنویسید که ۲ عدد از کاربر دریافت کند و خروجی کوتاه تفاضل، حاصلضرب و حاصل تقسیم را محاسبه کرده و نمایش دهد.

Ex : برنامه‌ای بنویسید که خروجی زیر را چاپ کند

```
$  $  $  $
$  $  $  $
$  $  $  $
```

```
#include(iostream.h)
#include (conio.h)
using name space std;
int main()
int $;
Cin >> $;
Cout >> "$ $ $/n $ $ $/n $ $ $ $";end;
```

Ex : برنامه‌ای بنویسید که ضرایب معادله درجه دوم را از ورودی بگیرد و معادله را حل کند

```
#include(iostream.h)
#include(conio.h)
using namespace std;
int main main()
int a,b,c,d,x1,x2;
Cout << "Enter 3 numbers(a,b,c) for ax^2+bx+c" << endl;
Cin >> a >> b >> c;
d = a*b-4*a*c
IF(d>0)
x1 = ((-b)-sqrt(d))/2*a)        return 0
x2 = ((-b)+sqrt(x))/2*a
else if(d==0)
x1 : (-b)/2*a
Cout << "x1,x2" << endl;
else if (d<0)
cout << "No answer - << end1;
```

MICRO

EX : اعداد زوج تا صد را چاپ کنید.

```
#include (iostream.h)
#include (conio.h)
using name space std;
int main() //int i;
for (i=0; i<100; i++)
cout << "i" <<;
getch
return 0
```

EX : 10 عدد صحیح از ورودی بگیرید مجموع و میانگین آنها را حساب کنید.

```
#includ (iostream.h)
#include(conio.h)
using name space std;
int sum; n; i; sum = 0;
for(i=0; I<10; I++);
{ cin >>
    sum = sum + n
    }
cout << "sum" << endl;
cout << "sum/10" <<
```

Subject: Date:

آرایه : تعدادی مشخص از خانه‌ها که پشت سر هم می‌باشند و همگی دارای یک نام و نوع یکسان هستند؛ به هر کدام از این خانه‌ها عنصر آرایه می‌گویند جهت دسترسی به هر عنصر آرایه نام آن به همراه شماره اندیس عنصر را فراخوانی می‌کنیم پس هر عنصر آرایه باید یک اندیس کلیدا داشته باشد به همین دلیل آرایه را متغیر اندیس دار نیز می‌گویند.

به عبارت آرایه دنباله‌ای از عناصر همسان می‌باشد. *

← جهت تعریف آرایه باید نام، نوع و طول آرایه را مشخص کنیم برای مثال:

تعریف آرایه‌ای با طول ۶ که int آن int a[6]

پس از آن مقداردهی به آرایه را بررسی می‌کنیم.

ممکن است مقداردهی در زمان تعریف برنامه یا در طول برنامه باشد. ؟

مقدار اندیس یا مقدار طول آرایه می‌تواند یک عبارت محاسباتی باشد به عنوان مثال در تعریف int[a+b] = 4 + مقدار اندیس برابر حاصل a+b می‌باشد.

* توجه داشته باشید که شماره گذاری عناصر آرایه همواره از ۰ شروع می‌شود؛ پس اگر عنصر nام از آرایه فراخوانی شود منظور دسترسی به خانه‌ی n-1 می‌باشد و آن به عنوان مثال a[n] منظور عنصری است که شماره اندیسش ۱ می‌باشد.

آرایه ممکن است یک بعدی باشد که به آن بردار می‌گویم و ممکن است دو بعدی باشد که به آن ماتریس می‌گویم. هم چنین می‌توانیم آرایه را به صورت چند بعدی نیز تعریف کنیم اما توجه داشته باشید که در نهایت همان مفهوم اولیه آرایه یک بعدی پیاده سازی می‌شود. به منظور مقداردهی آرایه در هنگام تعریف می‌توانیم به شکل زیر عمل کنیم همچنین اگر طول آرایه تعریف شده باشد و تنها لیستی از عناصر آرایه در دسترس باشد مانند: int num[ ] = {100, 50, 75, 122, 77, ∅}

compiler در زمان اجرا با توجه به عناصر طول آرایه را مشخص می‌کند همچنین اگر تعریف به صورت int num[10] = {∅} باشد آرایه‌ای به طول 10 را تعریف می‌کند که محتوای آن ۰ است اما اگر به صورت int num[10] = {2} باشد آرایه‌ای با طول 10 در نظر می‌گیرد که اولین عنصر آن ۲ و بقیه ۰ است.

```
int a[10] = {∅}
int a[10] = {2}
```

Subject: Date: 95.9.4

EX: برنامه‌ای بنویسید که ۲ آرایه‌ی ۱۰ تایی را از ورودی بگیرد حاصل جمع آن‌ها را در آرایه‌ی سوم ذخیره کند و خروجی دهد.

```
# include (iostream.h)
# include (conio.h)
using namespace std;
int main()
{const int =10
int a[n], b[n], sum[n]
for( int i=0 ; i<10 ; i++)
{ cout << " Enter first 10 numbers" <<;
  Cin>>a[n] ;
}
for(int i=0 ; i(10; i++)
{ cout << " Enter second number"
}Cin>> b[n] ;
for(i=0 ; i <10, i++)
{ sum[n]=a[n]+b[n];
  cout << sum[n];
}
  return 0
  endl;
```

EX: برنامه‌ای بنویسید که یک آرایه‌ی 10 عضوی از ورودی بگیرد سپس مجموع عناصر آرایه را محاسبه کند و در خروجی نمایش دهد.

```
const int n=10
int a[n], sum=0;
for(int i=0; i<n; i++)
{cout<<"plz enter number"<<\t<<i;
cin>>a[i];
cout<<a[i];
}
For(int i=0, i<n; i++)
{
sum+=a[i]
}cout<<"total is:"<<sum;
```

Ex: توسط آرایه عوارد مساوی برای ادراد {1,16,5,8,2,11,14,6,10}

```
int a[n]={ — }
coce+<<"number"<<\t<<"value"<<endl;
for(int i=0; i<n; i++)
{cout<<i<<\t<<a[i];
for(int j=0; j<a[i]; j++
cout<<" ";
}
ccut eul;
}
```

MICRO

Subject: / Date: 95.9.18

تابع : هدف از ایجاد تابع این است که برنامه نویس به جای داشتن یک برنامه ی بزرگ و یک پارچه ، یک برنامه ی ساخت یافته داشته باشد. در اینصورت برنامه به قطعات کوچکتری تقسیم می شود که هر کدام وظیفه ی خاصی را برعهده دارند. درک چنین ساختاری برای دیگران راحت تر بوده و اشکال زدایی آن نیز آسان تر است و همچنین به دلیل امکان تعریف متغیرهای محلی امکان تداخل متغیرها کم تر می باشد.

با استفاده از تابع می توانیم قطعات مفید و قطعاتی که کارهای روتین را انجام می دهند یک بار نوشته و بارها در برنامه های دیگر مورد استفاده قرار دهیم. در C++ به عنوان یک برنامه ی ساخت یافته می توانیم از توابع استفاده کنیم.

متغیرهای محلی که در تابع تعریف می شوند بیرون از تابع معنا ندارند در نتیجه کدهای های تابع از یکدیگر مجزا می باشند و اطلاعات آن ها فقط از طریق ورودی های تابع و خروجی های آن می باشد. یک تابع همان است که بر حسب نیاز تعدادی آرگومان ورودی داشته باشد و می تواند خروجی داشته باشد.

در تعریف توابع سه جزء اصلی وجود دارد که عبارتند از: 1. نوع خروجی 2. نام تابع 3. آرگومان های ورودی

به منظور تعریف آرگومان های ورودی پس از نام تابع ابتدا نوع آن آرگومان و سپس نام آن را می آوریم؛ در نهایت بلاکی از دستوراتی که همان بدنه ی تابع می باشد آورده می شود تعریف تابع با { شروع و با } تمام می شود. لازم است که حتی از یک مقدار بازگشتی تابع را فراخوانی کنیم.

→ return 0 , return 5 (مساحت)

یک تابع ساده

```
int Function name (int a, int b)
{
==
return a;
}
```

(نوع خروجی تابع — اسم تابع — نوع — اسم — آرگومان های ورودی)

یک تابع همان است خروجی نداشته باشد که در اینصورت نوع خروجی آن با void عالش داده می شود؛ هم چنین ممکن است که بدنه ی از یک ورودی داشته باشد که با " , " از هم جدا می شود اما حداقل می تواند یک خروجی داشته باشد!

Subject. Date:

ورودی های تابع به دو صورت می باشند 1. فراخوانی از طریق (یا) مقدار call by value
2. فراخوانی از طریق مرجع call by refrence

در نوع اول مقدار آرگومان ورودی به تابع ارسال می شود و تغییری در مقدار ورودی در متغیر دیگری که متغیر محلی است ذخیره می شود تغییراتی که در این متغیر محلی رخ می دهد به تابع فراخوان (اصلی) منتقل نمی شود.

در نوع دوم آدرس آرگومان ورودی به تابع ارسال می شود و در نتیجه هر گونه تغییر به متغیر اختصاص داده می شود و تمام تغییراتی که روی آرگومان اعمال می شوند حقیقتاً به تابع اصلی بازگردانده می شوند.

ورودی های یک تابع می توانند از نوع آرایه باشند بدان منظور نام آرایه و تعداد عناصر آن را به عنوان ورودی ارسال می کنیم.

Ex: برنامه ای بنویسید که دو عدد را با استفاده از تابع از ورودی بگیرد و محیط و مساحت مستطیل را به دست آورد.

```
# include (iostream)
# include <conio>
using name space std;
int gets (int tol, int arz);
int getp (int tol, int arz);
int main ()
{int tol, arz, s, p;
 cout << "tolo arz ra vared konid";
 cin >> tol >> arz;
 cout << endl;
 p = getp(tol, arz);
 s = gets(tol, arz);
 Cout << "s: " << s << "p: " << p;
 return 0;
}
int gets(int tol, int Arz)
{return tol * arz;
}
int getp(int tol, int arz)
{return(2*(tol+Arz));}
```

MICRO

Subject: Date: 95.9.18

* قبل از فراخوانی تابع باید تابع تعریف شده باشد یا اینکه به صورت کلی معرفی شده باشد، منظور از معرفی به صورت کلی این است که سطر اول در تعریف تابع بدون ذکر آرگومان های ورودی نوشته شود. اگر یک تابع را در ابتدا تعریف نکرده باشیم در زمان فراخوانی با error مواجه می شویم.

پشته ساختمان داده ای است که به صورت معکوس رشته می شود و خصوصیت LIFO دارد یعنی آخرین عنصری که به آن درج شود (push) عملی عضوی است که از آن خارج می شود (pop)
اولین

کاربرد ← یکی از کاربردهای پشته ذخیره ی آدرس خانه بازگشت می باشد.

تمرین
EX:- برنامه ای بنویسید که n آرایه شامل نمرات دانشجو و تعداد واحدرس را به یک تابع بفرستد تا معدل محاسبه شود.

توابع بازگشتی: توابعی هستند که به طور مستقیم یا غیرمستقیم چندین بار خود را فراخوانی می کنند این توابع در زبان های سطح بالا مورد استفاده قرار می گیرند اما از نظر اجرا کند هستند معمولا از آنها به توابع در حل مسائلی استفاده می کنیم که ماهیت بازگشتی دارند مانند فاکتوریل

پیاده سازی ضرب فاکتوریل دری فیبوناچی