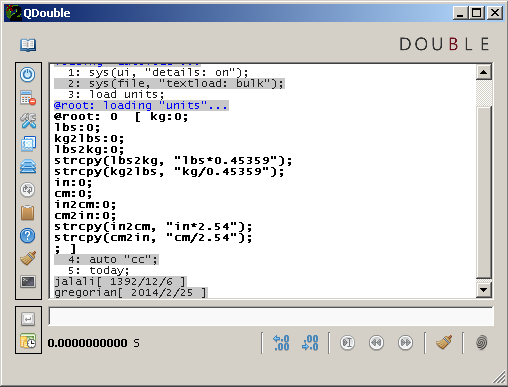
نرم افزار ماشین حساب مهندسی double

راهنمای کاربر

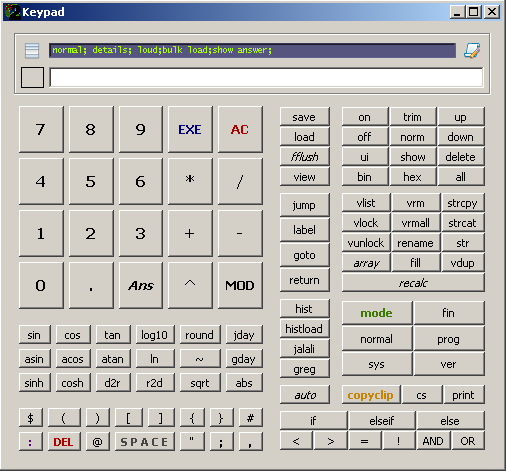
* **مقدمه:**

هدف از ساخت این نرم افزار ایجاد یک ماشین حساب مهندسی برای انجام امور روزمره دفتر فنی و صورت وضعیت بوده، بنحوی که قابل استفاده آسان بوده و بسیاری از عملیات ( اتوماسیون، متغیرها، توابع ریاضی، منطق، فایل ها و چند وظیفگی ) را پشتیبانی نماید، نرم افزار دارای دو نسخه با رابط گرافیکی و کنسول میباشد، آخرین نسخه نرم افزار از نشانی <http://sites.google.com/site/trailofamadman> قابل دریافت میباشد، همچنین مطالعه بخش [نکات حین استفاده](#usage_notes) در همین راهنما توصیه میگردد.

* **شمای نرم افزار( رابط گرافیکی ):**

****

عملیات مورد نیاز را در پنجره ورودی اطلاعات وارد نموده و پس از فشردن کلید [ Enter ] نتیجه را در پنجره خروجی ببینید، همچنین در حالت گرافیکی یک صفحه کلید مجازی نیز موجود میباشد.



(تصویر شماتیک بوده و با تغییر نسخه گرافیکی تغییر خواهد نمود)

برخی از قابلیتهای خاص نسخه گرافیکی (Cute) عبارتند از:

* حرکت بین تاریخچه عبارات محاسباتی با کلیدهای جهت بالا [ ▲ ] و پایین [ ▼ ].
* اجرای عبارت ورودی با دو بار فشردن کلید [ Space ] بصورت متوالی.
* تکمیل نام متغیرها، توابع و فرمانها با ورود بخشی از نام و فشردن متوالی کلید [ Tab ].
* پاک شدن پنجره ورودی با فشردن کلید [ Esc ].
* انتقال مکان نما از پنجره خروجی به ورودی با فشردن کلید [ Tab ].
* حالت برنامه نویسی با فشردن کلید] [ F2.
* حرکت بین پنجره خروجی، ورودی برنامه نویسی و ورودی تک خطی با کلید های

▲ ] [ CTRL + و ▲ ] [ CTRL +

* ویرایشگر آرایه [ aedit(*array*) ]
* ترسیم نمودار [ plot(*array\_x, array\_y*) ]

برخی از قابلیتهای خاص نسخه کنسول (con)عبارتند از:

* اجرای محاسبه وارد شده بعنوان آرگومان برنامه.
* بارگذاری فایل وارد شده بعنوان آرگومان (double load *filename*).

همچنین ذکر این نکته ضروری میباشد که استفاده از نسخه گرافیکی آسانتر بوده و جهت استفاده روزمره توصیه میگردد اما نسخه کنسول پایداری و سرعت بیشتری داشته و جهت استفاده در موارد حساس مناسب تر میباشد، همچنین این نرم افزار برروی سیستم عامل لینوکس پایدارتر میباشد؛

**قالب اعداد:**

مبنای ده: 123, -123, 123.456, -123.456

مبنای شانزده: 0x1b

مبنای دو: 0b10101011

علمی: 1.2e3, -1.2e\-1, 1.2e\+2

* **عملیات پایه:**

علائم پایه شامل جمع (+)، تفریق (-)، ضرب (\*)، تقسیم (/)، توان (^)، باقیمانده (%)، پرانتز باز و بسته ( () )، نشانگر اندیس متغیر (آرایه) ( [] )، ‘و’ بیتی ( & )، ‘یا’ بیتی ( | ) و تعریف متغیر یا مقدار دهی به متغیر (=:) میباشند.

جهت انجام یک عملیات پایه ریاضی کافیست عبارت را بصورت متعارف وارد نمائید، بعنوان مثال: 1+1؛ سپس کلید [Enter] را فشار دهید، عبارت از پنجره ورودی حذف و نتیجه را در پنجره خروجی مشاهده خواهید نمود. آخرین پاسخ همیشه توسط متغیر ans قابل دسترسی خواهد بود.

عملیات انجام پذیرفته به ترتیب اولویت عملگرهای ریاضی خواهد بود و پرانتز جهت اولویت دادن به یک عملیات بکار میرود، بعنوان مثال حاصل 2\*1+1 برابر 3 و حاصل 2\*(1+1) برابر 4 خواهد بود.

نرم افزار شامل میانبرهایی جهت تسریع در محاسبات میباشد، مثلا" 1+ آخرین پاسخ را بعلاوه 1 نموده و حاصل را در متغیر ans ذخیره خواهد نمود، این میانبر شامل تمامی عملیات پایه میباشد.

همچنین نرم افزار قابلیت انجام چندین محاسبه در یک خط ورودی را نیز دارا میباشد، جهت انجام اینگونه محاسبات کافیست قطعات مختلف را بوسیله ; از یکدیگر جدا نمائید، مثلا" پاسخ عبارت 1+1;+1; برابر 3 خواهد بود.

جهت درج مقادیر منفی کافیست یک نماد منفی پشت عدد قرار بدهید؛ مثلا" 1-+1 برابر صفر میباشد، این میانبر جهت متغیرها نیز قابل اعمال باشد، بعنوان مثال ans--1 برابر (ans-)-1 میباشد و عباراتی مانند 1-\*1 قابل قبول میباشند، جهت استفاده از اعداد منفی بعنوان اولین عبارت جهت جلوگیری از تداخل با میانبر ‘-‘ آنرا در پرانتز قرار دهید، مثلاً عبارت -1 در ابتدای جمله برابر با ans-1 میباشد در حالیکه حاصل عبارت (-1) منفی یک میباشد.

همچنین اگر بدون تایپ عبارت تنها کلید [Enter] را بفشارید آخرین عبارت اجرا میشود؛ مثلا تایپ کنید 1+ و چندین بار کلید [Enter] را بفشارید.

جهت قراردادن یک عبارت بعنوان توضیح و جلوگیری از اجرای عبارت کافیست علامت # را به ابتدای آن بیفزایید، مثلا" حاصل عبارت 3+;2+#;1 برابر 4 میباشد.

1. **عملگرهای منطقی:**

شامل عملگرهای بزرگتر از (<)، کوچکتر از (>)، مساوی با (==)، کوچکتر یا مساوی با (=>)، بزرگتر یا مساوی با (=<)، نامساوی با (=!)، ‘و’ منطقی (&&) و ‘یا’ منطقی (||) میباشند؛ این عملگرها عمدتا در عبارات منطقی و همراه با توابع if و elseif بکار برده میشوند.

بعنوان مثال: اگر a برابر 1 و b برابر 2 و c برابر 3 باشد، حاصل عبارت

If(a>1||b<c){1;} else {2;}

برابر با 1 خواهد بود؛ جهت اطلاعات بیشتر به بخش توابع منطقی رجوع نمایید.

1. **عملگرهای ویژه:**
   1. عملگر #: جهت تعریف یک عبارت بعنوان توضیح و جلوگیری از اجرای آن.
   2. عملگر @: جهت دسترسی به متغیرهای موجود در پردازش پس زمینه و بصورت *@parent@child@variable*، هر فرزند تنها به متغیرهای فرزند خود دسترسی دارد و دسترسی به متغیرهای والد برای فرزند ممکن نیست.
   3. عملگر $: جهت خروج از حالت برنامه نویسی.

* **متغیرها:**

متغیرهای ساده با عملگر : قابل معرفی به برنامه میباشند؛ اگر متغیر به برنامه معرفی نشده باشد با این عملگر در حافظه ایجاد میگردد و در صورت وجود متغیر در حافظه مقدار آن به مقدار جدید تغییر خواهد نمود اما تعریف آن در حافظه برابر با تعریف اولیه باقی خواهد ماند (این تعریف با vexp قابل تغییر میباشد.)، باقی بودن تعریف متغیر باعث افزایش انعطاف پذیری آن میگردد (تابع [recalc](#recalc))؛ جهت روشن تر شدن موضوع به چند مثال میپردازیم:

a:=1 تعریف و مقدار متغیر a را برابر 1 قرار میدهد.

B:=a+1 تعریف b را برابر a+1 و مقدار آنرا برابر 2 قرار میدهد.

a:=2 فقط مقدار a را برابر 2 قرار میدهد.

re(b) مقدار b را دوباره حساب نموده و برابر 4 قرار میدهد.

re(a) مقدار a را دوباره برابر 1 قرار خواهد داد.

همچنین روش دیگری نیز برای استفاده از متغیر ها در ترکیب با تابع [recalc](#recalc) موجود میباشد، شما میتوانید یک متغیر را بصورت بلوک معرفی نموده و از آن مانند یک ماکرو استفاده کنید، کافیست تابع را بصورت variable:={expression} معرفی نموده و هر بار جهت محاسبه عبارت داخل آکولادها تابع [recalc](#recalc) را فراخوانی نمایید، بعنوان مثال a:={b+1;print(ans);} ش را بصورت عبارت درج شده معرفی نموده و هر بار فراخوانی [recalc](#recalc) مقدار b+1 را محاسبه نموده و مقدار را برروی صفحه نمایش چاپ میکند.

همچنین انواع متغیر های آرایه ای تک بعدی و دو بعدی (ماتریس) نیز موجود میباشند که توسط فرمان array و یا عملگر := قابل تعریف بوده و اعضای این نوع متغیر ها با عملگرهای [] برای نوع یک بعدی و یا [][] برای انواع ماتریسی قابل دسترسی هستند، برای مثال فرمان array(a[3]) یا a[3]:=0; یک آرایه یک بعدی با سه عضو تعریف نموده و مقدار اولیه صفر را به همه تخصیص میدهد، این اعضا با عملگرهای a[0]، a[1] و a[2] قابل دسترسی خواهند بود. فرمان array(a[3], 5, 6) عملیات قبل را انجام میدهد و مقادیر 5 و 6 را به آرایه های a[0] و a[1] به ترتیب تخصیص میدهد؛ همچنین عبارت a[3]:=1;یک آرایه با سه عضو تعریف نموده و به a[2] مقدار 1 و به بقیه مقدار 0 را تخصیص میدهد.

به همین ترتیب فرمان array(a[2][3]) یا عبارت a[2][3]:=0; یک ماتریس 3×2 را تشکیل داده و به همه اعضا مقدار صفر را تخصیص میدهد، اعضا از a[0][0] الی a[1][2] شماره گذاری میگردند، همچنین array(a[2][3], 1, 2, 3) عملیات قبلی را با تخصیص مقادیر 1 و 2 و 3 به ترتیب بهa[0][0] ، a[0][1] و سپس a[1][0] انجام میدهد.

اندازه آرایه ها با تعریف مجدد توسط تابع array و یا عملگر := قابل افزایش می باشد، در این حالت مقادیر تعریف شده قبلی باقی خواهند بود.

متغیر ها میتوانند محافظت شده باشند، در اینصورت هیچگونه اجازه تغییر در آنها تا زمانیکه محافظت میگردند صادر نخواهد شد، جهت اطلاعات بیشتر به بخش *توابع کنترل متغیر* مراجعه نمایید.

برخی از متغیرها در سیستم رزرو هستند، کنترل این متغیرها بعهده برنامه میباشد و عبارتند از ans جهت نگهداری نتیجه آخرین عملیات محاسباتی، pi، e و clip جهت دسترسی به محتویات کلیپبورد.

* **توابع ریاضی:**

شامل توابع مثلثاتی sin, cos, tan و توابع arc مانند asin و توابع hyperbolic مانند cosh؛ توابع لگاریتم log و ln؛ توابع تبدیل radtodeg و degtorad؛ تابع جذر sqrt؛ تابع قدر مطلق abs؛ تابع جزء صحیح intval، توابع گرد نمودن عدد round و ~ و توابع تبدیل تاریخ به عدد jday و gday میباشد.

* *توابع مثلثاتی:*

1. sin: جهت باز گرداندن مقدار سینوس یک عدد یا متغیر بکار میرود، آرگومان ورودی برحسب درجه میباشد؛ مثلا" sin(90) برابر 1 میباشد.
2. cos: تابع کسینوس.
3. tan: تابع تانژانت.
4. asin: تابع آرک سینوس.
5. acos: تابع آرک کسینوس.
6. atan: تابع آرک تانژانت.
7. sinh: تابع سینوس هیپربولیک.
8. cosh: تابع کسینوس هیپربولیک.

* *توابع لگاریتمی:*

1. log یا log10: لگاریتم پایه 10 یک مقدار؛ بعنوان مثال log(10) یا log10(ans).
2. ln: لگاریتم نپرین (پایه e)

* *توابع تبدیل:*

1. radtodeg یا r2d: تبدیل یک مقدار برحسب رادیان به درجه، بعنوان مثال r2d(pi) برابر 180 میباشد و cos(r2d(pi)) برابر 1- است.
2. degtorad یا d2r: تبدیل درجه بر رادیان.

* *تابع جذر:*

1. sqrt یا sqr: ریشه دوم یک مقدار را محاسبه مینماید.

* *تابع قدر مطلق:*

2- abs: قدر مطلق یک مقدار را محاسبه مینماید.

* *توابع گرد کردن:*

1. round: جهت گرد کردن اعداد اعشار بکار میرود و شکل کلی آن بصورت

round(*number*, *precision*, *mode*) میباشد؛ آرگومان اول عدد ورودی، آرگومان دوم دقت اعشار و آرگومان سوم نحوه گرد شدن عدد میباشند. آرگومان سوم میتواند یکی از مقادیر up جهت گرد شدن به سمت بالا، down یا dn جهت گرد شدن به سمت پایین، trim یا tr جهت برش مقدار از محل دقت اعشار مورد نیاز و normal یا nr جهت گرد شدن به این صورت که اگر رقم بعد از رقم اعشار مورد درخواست بزرگتر یا مساوی پنج باشد رقم به سمت بالا و در غیر اینصورت به سمت پایین گرد خواهد شد. این تابع با دو و یک آرگومان نیز قابل استفاده میباشد، به این صورت که پیش فرض آرگومان دوم مقدار 2 و پیش فرض آرگومان سوم normal میباشد؛ بعنوان مثال:

round(90.12, 1, trim) مقدار 90.1 را باز میگرداند؛ round(90.125,2) و round(90.125) هردو مقدار 90.13 را باز میگردانند؛

1. ~: این تابع معادل با round(number, 2, normal) یا همان round(number) میباشد.

* *توابع محاسبه تاریخ:*

1- jday: شماره روز تاریخ تقویم شمسی را باز میگرداند، این تابع جهت محاسبه فاصله زمانی بین دوتاریخ کارامد میباشد، بعنوان مثال jday(13920410)-jday(“1392/4/9”) مقدار 1 را باز میگرداند؛

2- gday: شماره روز تاریخ تقویم میلادی را باز میگرداند، این تابع و تابع بالا همیشه بایستی با تابع تقویم همنوع خود بکار گرفته شوند، وجود gday و jday در یک عبارت بعلت همسان نبودن شماره روز میلادی و شمسی نتیجه درستی نخواهد داشت، بهتر است ابتدا تاریخهای مورد نظر خود را به یک تقویم واحد برگردانید؛ ر.ج. jalali و greg.

* **توابع اسکریپت نویسی:**

شامل توابع تصمیم گیری if، else، elseif؛ توابع ارتباط با کیلپبورد copyclip و copyclipsep؛ توابع کنترل متغیر vrm, vrmall, vdesc, vexp, vlist, vstr, vlock, vunlock, array, afill, recalc, ؛ توابع ارتباط با فایل load, save؛ توابع label و goto، تابع returnو jump و تابع اتوماسیون autorun؛ توابع دسترسی به تاریخچه history, histload و توابع تبدیل تاریخ hijrj و greg.

نکته قابل توجه در استفاده از این نوع توابع استفاده آنها در عبارات مجزا جهت عملکرد صحیح آنها میباشد، مثلا" vrm(a);vlist;، همچنین در توابعی که آرگومان آنها بصورت لیست میباشد آرگومانها با یک کاما (,) از یکدیگر جدا میشوند، مثلا" vrm(a,b,c).

* *توابع تصمیم گیری:*

1. if: این تابع بصورت if(*logical\_expression*){*expression*} استفاده میگردد، منظور از عبارت منطقی عباراتی مانند 1<2 (صحیح) ، 1+1>2&&3<4 (غلط) میباشند، در صورت صحیح بودن عبارت داخل پرانتز، عبارت داخل آکولادها اجرا میشود، در صورت غلط بودن اگر تابع elseif وجود داشته باشد صحت عبارت منطقی آن بررسی شده و باقی رفتار آن مانند تابع if میباشد؛ اگر هیچیک از عبارات if و یا elseif صحیح نبوده و یا elseif وجود نداشته باشد و اگر else موجود باشد عبارت داخل آکولاد مربوط به else اجرا میگردد، در غیر اینصورت باقی عبارات جمله اجرا خواهند شد، مثال:

if(a<2&&b>2) {1;a:3;} elseif (c=a|b!=3) {0;+1;} else {+2;}

1. elseif: مانند if استفاده میگردد و در صورت عدم صحت عبارت if، صحت عبارت مربوطه بررسی میگردد؛ جهت راهنمایی بیشتر به if مراجعه نمایید.
2. else: بصورت else {*expression*} استفاده شده و فاقد عبارت منطقی میباشد، به فاصله موجود بین else و آکولاد توجه نمایید، این فاصله الزامیست؛ جهت راهنمایی بیشتر به if مراجعه نمایید.

* *توابع ارتباط با کلیپبورد:*

1. copyclip یا cc: این تابع جهت کپی یک مقدار به کلیپبورد سیستم استفاده میشود و صورت کلی آن بشکل cc(*expression*) میباشد. آرگومان expression میتواند یک رشته، متغیر، عدد و یا عبارت محاسباتی باشد؛ اگر بعنوان مثال اگر متغیر hello برابر 1000 باشد، فرمان cc(hello) مقدار 1000 را به کلیپبورد کپی مینماید اما فرمان (cc("hello" کلمه hello را به کلیپبورد کپی خواهد نمود؛ همچنین اگر goodbye بعنوان متغیر تعریف نشده باشد، فرمان cc(*goodbye*) کلمه goodbye را کپی خواهد نمود؛ اجرای این تابع بدون آرگومان باعث کپی مقدار متغیر ans در حافظه خواهد گردید و نیز اجرای این تابع با آرگومان fin باعث کپی خروجی حسابداری به کلیپبورد میگردد.
2. copyclipsep یا cs: این تابع دقیقا مانند تابع copyclip عمل مینماید با این تفاوت که مقدار عددی کپی شده بصورت دسته های سه رقمی تفکیک خواهد گردید.
3. num2text یا n2t: این تابع یک مقدار عددی را به حروف برگردانده و رشته حاصل را در کلیپبورد سیستم کپی می نماید، اجرای این تابع بدون آرگومان باعث اجرای عملیات برروی مقدار متغیر ans میگردد، نحوه استفاده بصورت num2text(*value*) می باشد.

* *توابع کنترل متغیر:*

1. recalc یا re: این تابع باعث محاسبه مجدد متغیر با توجه به عبارت تخصیص داده شده به آن میگردد و بصورت re(*variable\_list…*) یا re;استفاده میگردد. مثلا اگر بعنوان تعریف متغیر a داشته باشیم a:b+1 ، هربار اجرای فرمان re(a) باعث محاسبه مجدد مقدار متغیر a با توجه به مقدار کنونی b خواهد گردید و در واقع متغیر a مانند یک تابع بسیار ساده عمل خواهد نمود؛ اجرای این تابع بدون آرگومان باعث محاسبه مجدد تمامی متغیرهای تعریف شده و بدون قفل میگردد و همچنین استفاده از نام آرایه جهت محاسبه تمامی اعضای آرایه یا یک سطر بصورت re(*array[i]*) یا یک ستون بصورت re(*array[-1][j]*) میگردد.
2. vrm: این تابع باعث حذف متغیر یا آرایه از حافظه میگردد و بصورت vrm(*variable\_list…*) یا vrm all;جهت حذف کلیه متغیرهای موجوداستفاده میگردد.
3. vdesc: این تابع جهت ثبت، مشاهده، تغییر و یا حذف توضیحات یک متغیر بجز متغیرهای محافظت شده میگردد و بصورت vdesc(*variable*,"*description*"/*command*) ظاهر میگردد؛ آرگومان اول نام متغیر و الزامی میباشد، آرگومان دوم در صورت وجود شرح بر متغیر یا فرمان میباشد؛ فرمانها شامل show جهت نمایش شرح کنونی و remove جهت حذف شرح کنونی متغیر میباشند؛ در صورت عدم وجود آرگومان دوم شرح کنونی متغیر نمایش داده خواهد شد.
4. vexp یا strcpy: این تابع جهت تعریف عبارت تخصیص داده شده به متغیر جهت استفاده در تابع recalc استفاده میگردد و بصورت vexp(*variable*, "*expression*") میباشد، مثلا" strcpy(var1, str(var2));.
5. vcat یا strcat: این تابع جهت ترکیب دو رشته با هم میباشد و نحوه استفاده از آن مانند strcpy میباشد.
6. vlist: این تابع جهت مشاهده لیست متغیرهای تعریف شده مشاهده میگردد و بصورت vlist(*variable*/*array\_name*) یا بدون آرگومان استفاده میگردد؛ همچنین نام متغیر با جایگزینهای \* و ? و درون " بصورت "\**part\_of\_variable\_name*??" قابل استفاده میباشد.
7. vstr یا str: این تابع جهت بازگرداندن عبارت تخصیص داده شده به متغیر و بصورت vstr(*variable*) استفاده میگردد.
8. vstrcmp یا strcmp: این تابع جهت مقایسه مقدار دو رشته یا مقدار عبارت متغیر و رشته بکار میرود، نحوه استفاده: strcmp(*variable/string, variable/string*). مثال: if(strcmp(var, “+1”)!=1){print(“not plus one”);}.
9. vlock یا vlk: این تابع جهت مشاهده لیست متغیرهای حفاظت شده و یا قفل نمودن یک متغیر استفاده میگردد و بصورت vlock(*variable\_list…*) و یا بدون آرگومان استفاده میگردد، در صورت استفاده بدون آرگومان لیست متغیرهای حفاظت شده نمایش داده خواهند شد و در صورت درج نام، متغیر قفل خواهد گردید.
10. vunlockیا vun: این تابع عکس عملیات انجام شده با تابع vlock را انجام میدهد و همانند آن استفاده میگردد.
11. vren یا rename: این تابع جهت تغییر نام متغیرها و بصورت rename(*old\_variable, new\_variable*) بکار میرود.
12. vdup یا copyvar: این تابع جهت ایجاد کپی از متغیر بکار میرود و بصورت vdup(*destination, source*) قابل اجرا میباشد، این تابع همچنین قابلیت کپی آرایه دو بعدی و تک بعدی بصورت کامل و بصورت سطری و ستونی را با فرمت های vdup(*dst, src[i]*) و vdup(*dst, src[-1][j]*) را دارا میباشد.
13. array: این تابع جهت تعریف متغیرهای آرایه ای استفاده میگردد و بصورت array(*variable[size],value\_list…*) استفاده میگردد. آرگومان اول نمایانگر نام آرایه میباشد، مقدار قرار گرفته شده در کروشه (اندیس) اندازه آرایه بوده و پس از آن لیست اختیاری مقادیر به ترتیب شماره اندیس درج میگردد، شماره اندیسها از 0 شروع شده و تا یک شماره قبل از size (size-1) ادامه خواهد داشت؛ جهت روشن شدن موضوع به مثال توجه نمایید:

اگر متغیر i حاوی مقدار 5 باشد فرمان array(a[i], 1, 2)آرایه a را با اندیسهای 0 الی 4 (5 عدد عضو) بوجود آورده و به a[0] مقدار 1، و به a[1] مقدار 2، و به a[2]، a[3] و a[4] مقداری را تخصیص نخواهد داد؛

جهت دسترسی به اعضای آرایه مقادیری مانند a[1+1]، a[variable]، a[+2] معتبر میباشند، همچنین فراخوانی این تابع با نام آرایه موجود و عدد سایز بیشتر از سایز موجود باعث افزایش اندازه آرایه می گردد.

1. fill: این تابع یک مقدار خاص را به تمامی اعضای یک متغیر آرایه ای تخصیص میدهد و بصورت fill(*array*, *value*)و یا fill(*array, exp*) استفاده میگردد، همچنین در مورد ماتریسها این تابع بصورت fill(*array[row], value*) جهت تخصیص یک مقدار به کلیه ستونهای سطر *row* و یا fill(*array[-1][col], value*) جهت تخصیص یک مقدار به کلیه سطرهای ستون *col* بکار میرود.

* *توابع ارتباط با فایل:*

1- load: این تابع جهت بارگذاری یک فایل محاسباتی به برنامه بکار میرود و بصورت load("*filename*") استفاده میگردد. خطوطی از فایل که با کاراکتر # شروع میگردند بعنوان شرح در نظر گرفته شده و نادیده گرفته میشوند.

2- save: این تابع جهت درج عملیات انجام شده به یک فایل استفاده شده و بصورت save("*filename*"/*command*) استفاده میگردد، در صورت درج نام فایل، فایل مورد نظر در صورت عدم وجود ایجاد شده و در صورت وجود باز شده و ورودی های جدید به انتهای فایل اضافه میگردند. فرمانها عبارتند از stop جهت توقف ذخیر سازی و resume جهت ادامه ذخیره سازی عبارات.

1. fflush: این تابع جهت دسترسی به داده های فایل در حال ذخیره بدون بستن نرم افزار استفاده میگردد، در حالت عادی در حین ذخیره سازی فایل داده ها مستقیما" برروی دیسک نوشته نمیشوند، به این دلیل که این کار منابع زیادی را اتلاف خواهد نمود، لذا داده ها تنها پس از خروج با استفاده از فرمان exit و یا پس از صدور این فرمان واقعا" برروی دیسک ذخیره میگردند. شما میتوانید از فرمان auto "fflush;"; نیز جهت ذخیره سازی پس از اجرای هر خط استفاده کنید.
2. view: جهت مشاهده محتویات فایل ها، مانند load استفاده میگردد.

* *توابع کنترل جریان:*

1. label: این تابع جهت برچسب گذاری بر روی یک قطعه بکار میرود، برچسب ها بعدا" با تابع goto قابل ارجاع خواهند بود، مثلا" i:0;label a;i:i+1;if(i<3){goto a;} تا زمانیکه مقدار متغیر i کوچکتر از 3 باشد مقدار 1 را به متغیر i اضافه مینماید.
2. goto: این تابع جهت ارجاع به برچسب های تعریف شده استفاده میگردد، نکته مهم در استفاده از این تابع این است که برچسب ها قبل از استفاده از فرمان باید تعریف شده باشند.
3. return: این تابع جهت بازگشت از قطعه کد در حال اجرا استفاده میگردد، مثلا" if(i<3){return;} در صورتیکه مقدار متغیر i کوچکتر از 3 باشد اجرای کد حاضر را خاتمه میدهد.
4. jump: این تابع جهت نادیده گرفتن فرمانهای بعدی به تعداد درخواستی میگردد، بعنوان مثال حاصل 1;jump(2);+1;+1;+1; برابر 2 میباشد؛ همچنین فراخوانی این تابع بدون آرگومان jump;، معادل jump(1); میباشد.
5. fork: این تابع جهت اجرای یک عبارت در پس زمینه بکار میرود؛ پس از استفاده از این تابع شما قادر به انجام سایر محاسبات خود خواهید بود در حالیکه نرم افزار در حال انجام محاسبات در پس زمینه میباشد، همچنین شما با عملگر @ قادر به دسترسی به مقدار کنونی متغیر موجود در پردازش پس زمینه خواهید بود، نحوه استفاده از این تابــــع بصـــورت fork; fork(*name*, “*expression*”); و یا; fork(*name*, *variable*) میباشد.
6. kill: این تابع جهت خاتمه به اجرای پردازش پس زمینه ایجاد شده توسط fork بکار میرود و بصورت kill( *fork\_name)*; یا kill all;میباشد.
7. switch: این تابع جهت جابجایی بین پردازشهای مختلف بکار رفته و بصـــــــورت switch(*fork\_name)*; قابل استفاده میباشد، همچنین جهت جابجایی به پردازش والد از switch parent; و جهت انتقال به پردازش ریشه از switch root; استفاده نمایید، اجرای این تابع بدون آرگومان باعث نمایش نام پردازش جاری میگردد.
8. break: این تابع جهت قطع پردازش جاری بکار میرود (پردازش های در حال اجرا قابل kill و یا switch نیستند) و بصورت break(*fork\_name)*; و یا break all; بکار میرود.
9. sleep: جهت تعلیق پردازش بمدت n میکرو ثانیه بکار میرود؛ sleep(*n*).
10. exec: جهت اجرای فرمان و بصورت exec(*file*, *argument\_list*…) بکار میرود، جهت جلوگیری از قفل شدن پنجره در حین اجرای فرمان با fork ترکیب کنید.

* *تابع اتوماسیون:*

1- تابع autorun یا auto: این تابع جهت اجرای خودکار یک عبارت خاص پس از پایان اجرای هر عبارت بکار میرود و بصورت auto("*expression*"/*command/variable*) بکار میرود؛ در صورت عدم درج آرگومان و اجرای auto به تنهایی، وضعیت موجود نمایش داده میشود، در صورت درج فرمان off، اجرای خودکار عبارت متوقف گشته و در صورت استفاده از فرمان on اجرای خودکار ادامه پیدا میکند؛ بعنوان مثال:

auto("+1;cc;")

باعث اضافه شدن عدد 1 به حاصل نهایی پس از اجرای یک عبارت شده و مقدار را در کلیپبورد کپی مینماید.

همچنین در صورت درج نام یک متغیر بعنوان آرگومان، عبارت تخصیص داده شده به متغیر بعنوان دستور auto در نظر گرفته خواهد شد.

* *توابع دسترسی به تاریخچه:*

1- history: نمایش تاریخچه عبارات محاسباتی.

2- histload: بارگذاری شماره خط درخواستی از لیست عبارات و اجرای آن پس از فشردن کلید [Enter].

* *توابع تبدیل تاریخ:*

1- jalali یا shamsi: جهت تبدیل تاریخ میلادی به شمسی،بصورت jalali(*numericdate*) و jalali (“*date*”) استفاده میگردد، مثلا jalali (20130409) یا jalali (“2013/4/9”).

1. greg یا miladi: جهت تبدیل تاریخ شمسی به میلادی، جهت طرز استفاده ر.ج. jalali.
2. today: جهت بازگرداندن تاریخ امروز.
3. jdate: جهت تبدیل شماره روز شمسی به تاریخ، مثال: jdate(151000).
4. gdate: جهت تبدیل شماره روز میلادی به تاریخ، جهت طرز استفاده ر.ج. jdate.

* *تابع کنترل:*

1- mode: این تابع جهت کنترل حالت پردازش و خروجی بکار میرود و بصورت mode(*operation\_mode\_list…*) بکار میرود، حالات پردازش عبارتند از nr یا normal جهت پردازش بصورت معمول، fin جهت تنظیم خروجی صفحه نمایش و فایل خاص امور حسابداری و prog جهت حالت برنامه نویسی، در این حالت هر خط پس از فشردن کلید [Enter] اجرا نمیگردد و تنها پس از وارد نمودن کاراکتر ; و سپس فشردن کلید [Enter] به تنهایی و بدون وارد نمودن هیچ عبارت دیگری کل عبارت وارد شده اجرا میشود، همچنین برای گرفتن لیست از برنامه وارد شده کافیست کلید [Enter] را بفشارید، نکته مهم در حالت prog این امر میباشد که در این حالت انتهای جملات بصورت خودکار با کاراکتر ; بسته نخواهد شد و انتهای جملات باید توسط کاربر و با کاراکتر ; بسته شود، همچنین جهت خروج از این حالت از یک کاراکتر $ استفاده نمایید، فرمان mode در اینحالت بی اثر میباشد. مثلا mode(fin) حالت پردازش را برای امور حسابداری تنظیم نموده و mode(nr) آنرا بحالت عادی باز میگرداند.

* *تابع خروجی:*

1- print: این تابع جهت چاپ مقادیر و عبارات بر صفحه نمایش و بصورت print(*argument\_list…*) بکار میرود، مثلا" print("answer is", " ", ans); باعث چاپ جمله answer is ، یک فاصله و سپس آخرین نتیجه میگردد؛

1. tr یا trim باعث رفتن به سرخط جدید میگردد.
2. hex باعث چاپ مقدار بعدی در مبنای 16 میگردد؛ مثال: print(hex, 27);
3. bin باعث چاپ مقدار بعدی بصورت باینری میگردد.
4. all باعث نمایش جزئیات بیشتر در مورد آرگومــان بعد میگردد، مثـــلا" print(all, array); جهت نمایش کلیه اعضای یک آرایه
5. plot: این تابع جهت ترسیم مقادیر دو آرایه بصورت نمودار استفاده میگردد، بعنوان مثال اگر x و y هریک آرایه ای از اعداد باشند، plot(x,y) باعث ترسیم مقادیر این دو آرایه در یک پنجره خروجی خواهد گردید.
6. aedit: جهت ویرایش اعضای یک آرایه، aedit(x) یا بدون آرگومان.

* **فرمانها:**

1. sys: این فرمان جهت تنظیم پارامترهای نرم افزار از قبیل اندازه بافرها بکار میرود.
2. ver: این فرمان جهت رویت و یا کنترل نسخه هسته نرم افزار استفاده میگردد.
3. help: این فرمان جهت ارائه توضیحات درباره نرم افزار یا توابع استفاده میشود.
4. exit: این فرمان جهت خروج از نرم افزار استفاده میشود.

* نکات حین استفاده:

1. نرم افزار در هنگام راه اندازی فایل autoload را بصورت خودکار بارگذاری مینماید.
2. برای ساختن عبارات شرطی و حلقه ها با if میتوان از عباراتی مانند:

0;label a;+1;if(ans==100000){jump;}; goto a;…

استفاده نمود، استفاده از عباراتی مانند:

0; label a;+1;if(ans<100000){goto a;};

علاوه بر پایین آوردن کارایی پایداری نرم افزار را نیز کاهش میدهند، علت در اینست که عبارت دوم شرط را 100000 مرتبه اجرا مینماید اما عبارت اول با 1 مرتبه اجرای شرط به نتیجه میرسد!

1. اگر در هنگام کار با فایلها (بخصوص در هنگام بارگذاری بصورت bulk) و یا هنگام دیدن خروجی ها بنظر میرسد که این موارد کامل پردازش نشده و یا ناقص هستند، اندازه بافرها را تنظیم کنید، از فرمانهای sys(buf, “exp: *n*”); جهت تنظیم مقدار بافر ورودی (واحد بایت) و فرمان sys(buf, “out: *n*”); جهت تنظیم بافر خروجی (نمایش) استفاده نمایید.

* **SYS:**
  + 1. sys(ui, “hilight: off/on/stat”);: جهت خامـــــــوش و روشن نمودن و اطلاع از وضعیت حالت سایـــــه - روشن خروجی.
    2. :sys(ui, “groupnum|numsep: off/on/stat”); جهت تنظیم جدا کننده هزارتایی اعداد.
    3. sys(ui, “decimal|decim: trim/fix/stat/*n*/+/-”);: جهت تنظیم دقت اعشار و نحوه نمایش آن.
    4. :sys(ui, “verbose|v: off/on/stat”); جهت تنظیم میزان خروجی توابع.
    5. :sys(ui, “answer|ans: off/on/stat”); جهت تنظیم نمایش پاسخ.
    6. :sys(ui, “details|det: off/on/stat”); جهت تنظیم میزان اطلاعات محاسباتی نمایش داده شده.
    7. sys(ui, “locale: fa/en/stat”);: جهت تنظیمات محلی (انگلیسی / فارسی).
    8. sys(ui, “nerd: off/on”);: جهت تنظیم نمایش اطلاعات جالب!
    9. :sys(file, “textload: line/bulk/stat”); جهت تنظیم نحوه بارگذاری فایلهای متنی.

1. sys(buf, “exp/tok/out/hist: stat/*n*/+/-“);: جهت تنظیم اندازه بافر ها.

بابک آخوندی، بهار 1393 < jaysi13@gmail.com >