✓کارنیل، بزرگترین شبکه موفقیت ایرانیان می باشد، که افرادی زیادی توانسته اند با آن به موفقیت برسند، فاطمه رتبه ۱۱ کنکور کارشناسی، محمد حسین رتبه۶۸ کنکور کارشناسی، سپیده رتبه ۳ کنکور ارشد، مریم و همسرش راه اندازی تولیدی مانتو، امیر راه اندازی فروشگاه اینترنتی،کیوان پیوستن به تیم تراکتور سازی تبریز، میلاد پیوستن به تیم صبا، مهسا تحصیل در ایتالیا، و..... این موارد گوشه از افرادی بودند که با کارنیل به موفقیت رسیده اند، شما هم می توانید موفقیت خود را با کارنیل شروع کنید.

برای پیوستن به تیم کارنیلی های موفق روی لینک زیر کلیک کنید.

www.karnil.com

همچنین برای ورود به کانال تلگرام کارنیل روی لینک زیر کلیک کنید.

https://telegram.me/karnil

عنوان: آموزش نرم‌افــزار GPSS

درس شبیه‌سازی کامپیوتر

استاد مربوطه: جناب آقای محمدیان سمنانی

گردآورندگان:

سید محسن میرحسینی [۱]

امیر عزیزخانی روشن [۲]

محمدرضا رازیان [۳]

دانشگاه سمنان

آذر ۹۰

۱- دانشجوی مهندسی نرم‌افزار. ایمیل: ssiedmohsenmirhossieny@yahoo.com

۲- دانشجوی مهندسی نرم‌افزار. ایمیل: amir.azizkhani.ros@gmail.com

۳- دانشجوی مهندسی فناوری اطلاعات. ایمیل: razian.mr@gmail.com

از آن‌جایی که ممکن است برخی از خوانندگان با دستورات GPSS هیچ آشنایی نداشته باشند، قسمت ذیل آورده شده است.

در صورت آشنایی می‌توانید به صفحه ۳ بروید.

آشنایی ابتدایی با دستورات GPSS

GPSS اصولا برای شبیه سازی سیستم‌های گسسته (به خصوص صفی) طراحی شده و دارای نوشتارهای (Version) متعددی است و معمولاً برای شبیه‌سازی از تکنیک پردازش فرایندها استفاده می‌کند. هر یک از دستورات GPSS یک فعالیت از سیستم را شبیه سازی می‌کند و شامل یک مجموعه پارامترهای ورودی می‌باشد. در مدل‌های GPSS اشیاء یک سیستم به دو دسته متمایز از یکدیگر تقسیم می‌شوند که آن‌ها را اشیاء سرویس دهنده و سرویس گیرنده می‌نامیم و یا به عبارت دیگر دسته اول را Facility (وسیله) یا انباره (Storage unit) و دسته دوم را اشیاء موقت یا (Transaction) می‌نامیم. البته تفاوت وسیله و انباره این است که وسیله در یک زمان توسط یک شیء موقت اشغال می‌شود در حالی که انباره به وسیله چند شیء موقت (تا حد گنجایش) به طور همزمان اشغال می‌گردد.

الف) اشیاء موقت

دستور GENERATE A,B,C,D,E

عناصر یا یا اشیاء ورودی به سیستم را در مدل تولید می‌کند.

A : متوسط زمان بین تولید (ورود به مدل) که اگر ذکر نگردد به طور فرضی صفر در نظر گرفته می‌شود.

B : پارامتر اصلاحی زمان بین تولید است اگر B مقداری ثابت باشد آن گاه زمان تولید بین دو شی موقت تصادفی و دارای توزیع یکنواخت در فاصله (A-B,A+B) است. واضح است که اگر B ذکر نگردد و یا صفر باشد زمان بین تولید مقدار ثابت A خواهد بود. و اگر به جای B نام یک تابع بیاید آن‌گاه زمان بین تولید اشیاء حاصل‌ضرب A در مقدار تابع است.

C : فاصله زمانی تولید اولین شیء از ابتدای شبیه سازی است. اگر C ذکر نگردد اولین شیء بعد از X واحد زمانی که X عددی است محاسبه شده توسط پارامترهای A و B تولید می‌گردد.

D : تعداد کل اشیاء موقتی که در طول شبیه سازی تولید می‌گردند در صورت ذکر نکردن آن، این عدد نا محدود است.

E : اولویت اشیاء تولیدی است. در صورت ذکر نشدن آن اشیاء دارای اولویت یکسان هستند.

توجه شود که اگر ذکر پارامتری لازم نباشد جای آن خالی نمی‌ماند و ویرگول‌ها پشت سر هم و بدون فاصله درج می‌شوند.

مثلاً دستور GENRATE 10,5,7,100

۱۰۰ شیء موقت با فاصله زمانی یکنواخت در فاصله (۱۵و۵) تولید می‌کند که اولین آن‌ها در واحد هفتم بعد از شروع شبیه سازی بوده و اولویت همه آن‌ها یکسان است.

با دستور TERMINATE A، A را از مدل حذف می‌کند.

برای تعیین یا تغییر یک مشخصه، از دستور ASSIGN A,B استفاده می‌شود که در آن A شماره مشخصه و B مقدار جدید آن است. مثلاً

ASSIGN 2,15

مقدار دومین مشخصه شیء وارد شونده را برابر ۱۵ قرار می‌دهد.

مثلا دستور ASSIGN A+,B مقدار B را به مقدار قبلی مشخصه A اضافه می‌کند. دستور ASSIGN A-,B مقدار B را از مقدار قبلی مشخصه A کم می‌کند.

ASSIGN 2+,15 ۱۵ واحد به دومین مشخصه شی ورودی اضافه می‌کند.

ب) وسایل، انباره‌ها و صف

دستور SEIZE A اشغال یک وسیله در صورت آزاد بودن وسیله A است. A نام آن وسیله است که می‌تواند نام سمبلیک یا شماره یک وسیله باشد.

پایان یافتن یک سرویس یا آزاد ساختن آن به وسیله دستور RELEASE A صورت می‌گیرد که A شماره وسیله یا نام سمبلیک وسیله آزاد شدنی است.

یک مجموعه سرویس‌دهنده موازی (سرویس‌دهنده‌هایی که سرویس‌های یکسان ارائه می‌کنند) را در GPSS انباره می‌نامند. گنجایش یک انباره همان تعداد سرویس‌دهنده‌ها است. در یک سیستم ممکن است یک یا چند انباره باشند که نام و گنجایش هر کدام باید تعیین شود. این عمل به وسیله A STORAGE name انجام می‌گیرد. A گنجایش انباره است.

برای ورود به یک انباره و اشغال یک یا چند واحد از گنجایش آن از دستور ENTER A,B استفاده می‌کنیم که A نام یا شماره انباره و B تعداد واحدهای اشغال شونده هستند. همچنین خروج از انباره A و آزاد ساختن B واحد از گنجایش آن از دستور LEAVE A,B استفاده می‌شود.

دستور ENTER در صورتی اجرا می‌شود که تعداد سرویس‌دهنده واحدهای آزاد انباره در این زمان بزرگتر مساوی B باشد و در غیر این صورت شی وارد شونده به انباره در دستور ما قبل ENTER به حالت انتظار تا آزاد شدن تعداد کافی از واحدهای انباره باقی می‌ماند .

سیستم شبیه‌سازی چندمنظوره GPSS (General Purpose Simulation System)

هدف از این آموزش تنها آشنایی با نرم‌افزار GPSS می‌باشد. در این آموزش طریقه شبیه‌سازی یک مسئله را با استفاده از نرم‌افزار GPSS ارائه می‌دهیم. پس از آن با برخی از امکانات GPSS هم آشنا می‌شویم. فرض بر این است که خواننده با درس شبیه‌سازی آشنا می‌باشد.

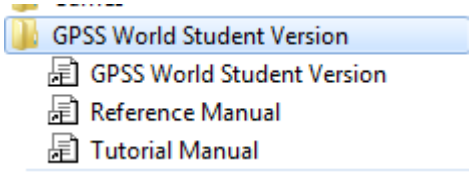
تذکر مهم: این آموزش یک منبع رسمی نمی‌باشد. مطالبی که در این آموزش بیان کرده‌ایم صرفاً برداشت ما از منبع اصلی آموزش GPSS (که در فهرست منابع ذکر گردیده) می‌باشد، لذا می‌تواند در آن خطا وجود داشته باشد. در صورت تمایل می‌توانید خطاهای موجود (خطاهای محتوایی) را به آدرس amir.azizkhani.ros@gmail.com نمایید. با تشکر.

نحوه نصب نرم‌افزار GPSS World Student Version 5.2.2

برای دانلود این نرم‌افزار می‌توانید به وب‌سایت http://www.minutemansoftware.com مراجعه نمایید. لازم به ذکر است که این ویرایش از GPSS (Student Version) رایگان می‌باشد (free of licensing fees).

نصب این نرم‌افزار بسیار آسان می‌باشد (یک سری Next !)، لذا آن را توضیح نمی‌دهیم.

نکته حائز اهمیت این است که به همراه نصب نرم‌افزار یک پوشه به نام GPSS World Student Version (شکل ۱) در Start -> Programs یا Start -> All Programs ایجاد می‌شود (البته اگر تنظیمات default را تغییر نداده باشید) که در آن فایل اجرایی برنامه و دو فایل دیگر با نام‌های Reference Manual و Tutorial Manual وجود دارند که آموزش‌های جامعی از نرم‌افزار GPSS می‌باشند.



شکل ۱

و دیگر اینکه اگر به مسیر نصب نرم‌افزار بروید یک پوشه به نام Minuteman Software ایجاد شده است. آن را باز کنید. پوشه GPSS World Student Version را نیز باز کنید. در اینجا یک پوشه با نام Samples وجود دارد که محتوی مثال‌های فراوانی است.

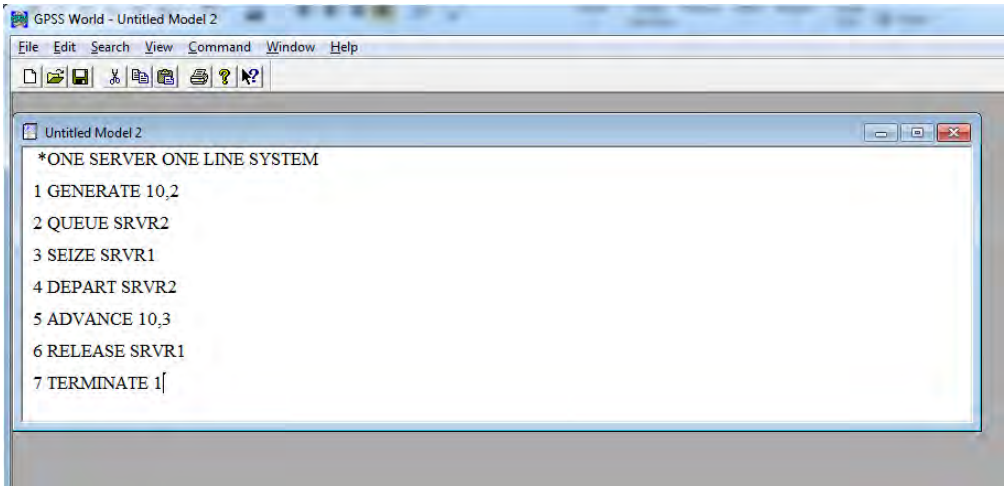
شروع کار با نرم‌افزار (Getting Started)

مثال ۱: یک سیستم ساده (یک صف-یک سرویس‌دهنده) را در نظر بگیرید.که زمان بین ورود مشتری‌ها دارای توزیع یکنواخت در فاصله (8,12) بوده و زمان انجام سرویس نیز تصادفی و یکنواخت در فاصله (7,3) باشد. مدل شبیه‌سازی این سیستم برای 250 مشتری به صورت زیر است که در آن نام سرویس‌دهنده SRVR1 و نام صف SRVR2 منظور شده است.

```
*ONE SERVER ONE LINE SYSTEM
1 GENERATE 10,2
2 QUEUE SRVR2
3 SEIZE SRVR1
4 DEPART SRVR2
5 ADVANCE 10,3
6 RELEASE SRVR1
7 TERMINATE 1
```

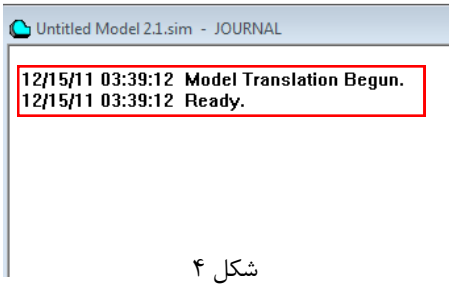
شکل ۲. برنامه مثال ۱

نرم افزار GPSS را اجرا کنید. از منوی File گزینه New را انتخاب کنید. در پنجره باز شده Model را انتخاب و بر روی دکمه OK کلیک نمایید. در حال حاضر یک پنجره باز شده است (به این پنجره، Model می‌گوییم)، دستورات شکل ۲ را copy و در این پنجره paste کنید. مانند شکل ۳.



شکل ۳

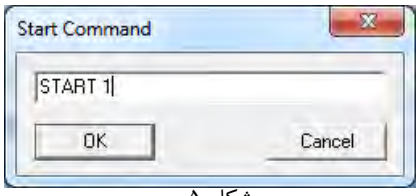
سپس از منوی Command بر روی گزینه Create Simulation کلیک کنید. پنجره‌ای نمایش داده می‌شود که نام آن Journal است (شکل ۴).



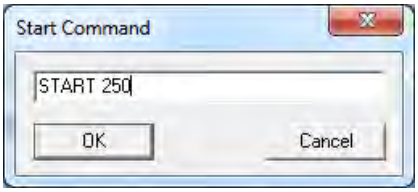
شکل ۴

در صورتی که پیغامی به غیر از این نمایش داده شده باشد (مثل ***Model Translation Aborted***) بررسی کنید که دستورات وارد شده دقیقاً مانند شکل ۲ می‌باشد. اگر دستوری را اشتباه وارد کرده‌اید آن را اصلاح کنید. سپس از منوی Command بر روی گزینه Retranslate کلیک کنید پنجره نمایش داده شده مانند شکل ۴ می‌باشد.

سپس از منوی Command بر روی گزینه START کلیک کنید. در پنجره بازشده (شکل ۵) به جای عدد ۱، عدد ۲۵۰ (تعداد مشتری‌ها یا تعداد Transaction ها) را وارد (شکل ۶) و بر روی دکمه OK کلیک نمایید.

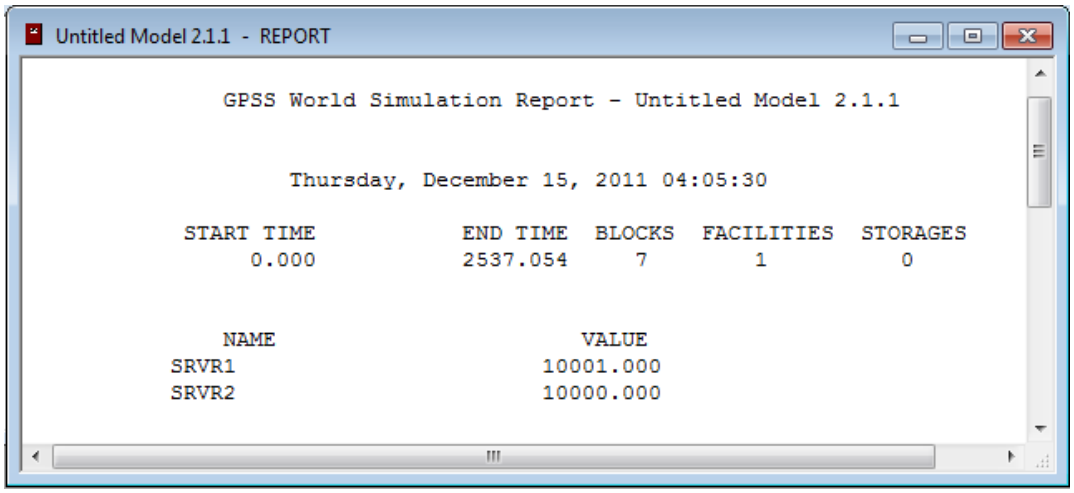


شکل ۵



شکل ۶

پنجره‌ای نمایش داده می‌شود که نام آن Report است (شکل ۷) در این پنجره گزارشی از شبیه سازی انجام گرفته، ارائه شده است.

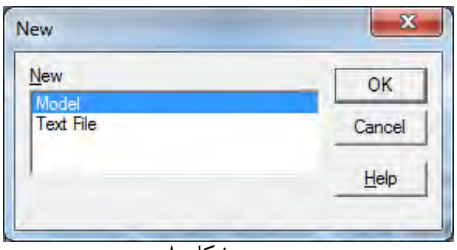


شکل ۷

چه اتفاقی افتاد؟!

روال کلی. مراحل اجرای یک مدل به ترتیب زیر می‌باشد. ابتدا باید یک مدل ایجاد کنید(File -> New/ Model). سپس باید دستورات را وارد مدل ایجاد شده نمایید. پس از اتمام ورود دستورات، نوبت به ترجمه (Translate) دستورات می‌باشد (Command -> Create Simulation). اگر برنامه خطا نداشته باشد یک Simulation Object از مدل نوشته شده، ایجاد می‌شود (زمانی که در پنجره Journal عبارت Ready نمایش داده شد یک Simulation Object ایجاد می‌شود). به طور معمول دستور START برای آغاز یک شبیه‌سازی (Simulation) استفاده می‌شود. در نتیجه از منوی Command گزینه START را انتخاب کرده و در پنجره باز شده به جای عدد ۱ تعداد Transaction های دلخواه را وارد می‌کنیم و OK می‌کنیم. زمانی که یک Simulation Object دستور START را اجرا می‌کند، Transaction ها در Simulation جریان می‌یابند. معمولاً شبیه‌سازی (Simulation) به طور خودکار پایان می‌یابد و یک گزارش تولید و نمایش داده می‌شود. البته اگر در حین اجرا Error رخ دهد، گزارشی ایجاد نمی‌شود.

روال جزئی. با نحوه ایجاد یک مدل اشنا شدیم. کافی است تا از منوی فایل گزینه New را انتخاب کنیم. در پنجره باز شده دو گزینه وجود دارد (شکل ۸): ۱– Model ۲– Text File



شکل ۸

کاربرد Model را قبلاً دیدیم. امّا در صورت نیاز به یک فایل متنی بر روی Text File کلیک می‌کنیم. با کلیک بر روی Text File صرفاً یک فایل متنی (.txt) ایجاد می‌شود.

انواع دستورات در GPSS

یک GPSS Model شامل یک دنباله از Model Statement ها می‌باشد. حال یک Model Statement شامل موارد زیر می‌باشد.

- Model Statement
  - GPSS Statement
    - Block Statement
    - Command Statement
  - PLUS Procedure

منظور از Model Statement تمامی دستوراتی است که می‌توانیم در یک Model (File->New->Model) بنویسیم. که این دستورات یا مربوط به GPSS اند و یا مرتبط با PLUS (Programming Language Under Simulation). درباره PLUS ها بعداً توضیح می‌دهیم. امّا دستورات GPSS هم به دودسته Block و Command تبدیل می‌شوند.

Block Statement: Block Statements to create GPSS Block Entities (تعریف موجود در Reference Manual). امّا شاید این طور بتوان گفت که دستورات بلوکی باعث ایجاد یک مانع می‌شوند. در نظر بگیرید در برنامه مثال ۱ مثلاً دستور SEIZE یک دستور بلوکی است. زمانی که یک Transaction توسط دستور GENERATE ایجاد می‌شود وقتی که به دستور SEIZE می‌رسد اگر سرویس‌دهنده مشغول باشد اجرای این Transaction در همین قسمت از کد متوقف می‌شود. یعنی مانع از اجرای ادامه کد می‌شود.

دستورات بلوکی عبارت اند از:

ADOPT, ADVANCE, ALTER, ASSEMBLE, ASSIGN, BUFFER, CLOSE, COUNT, DEPART, DISPLACE, ENTER, EXAMINE, EXECUTE, FAVAIL, FUNAVAIL, GATE, GATHER, GENERATE, INDEX, INTEGRATION, JOIN, LEAVE, LINK, LOGIC, LOOP, MARK, MATCH, MSAVEVALUE, OPEN, PLUS, PREEMPT, PRIORITY, QUEUE, MSAVEVALUE, OPEN, PLUS, PREEMPT, PRIORITY, QUEUE, READ, RELEASE, REMOVE, RETURN, SAVAIL, SAVEVALUE, SCAN, SEEK, SEIZE, SELECT, SPLIT, SUNAVAIL, TABULATE, TERMINATE, TEST, TRACE, TRANSFER, UNLINK, UNTRACE, WRITE

که با برخی از آن‌ها قبلا آشنا شدیم و با برخی دیگر نیز در ادامه آشنا می‌شویم.

Command Statement: که باعث ایجاد دستورات بلوکی نمی‌شوند. از این دستورات برای تعریف موجودیت‌ها (Entity) و کنترل شبیه‌سازی (Simulation) در حال اجرا استفاده می‌کنیم. Command ها ممکن است بخشی از دستورات نوشته شده درون مدل باشند و یا به عنوان دستور Interactive به Simulation موجود ارسال شوند (در مورد دستورات Interactive در ادامه صحبت می‌کنیم).

دستورات Command عبارت اند از:

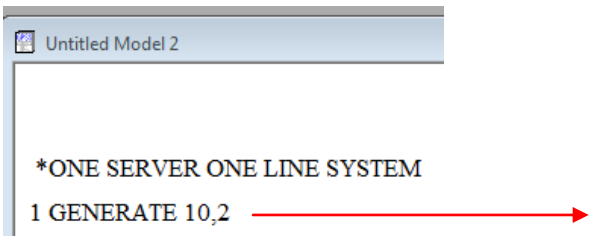
BVARIABLE, CLEAR, CONTINUE, EQU, EXIT, FUNCTION, FVARIABLE, HALT, INCLUDE, INITIAL, INTEGRATE, MATRIX, QTABLE, REPORT, RESET, RMULT, SHOW, START, STEP, STOP, STORAGE, TABLE, VARIABLE

ساختار GPSS Statement

هر GPSS Statement ترکیبی از چند فیلد می‌باشد.

Line number (optional) | Label (optional) | Verb (required) | Operands (depend on verb) | Comment (optional).

Line number : یا شماره خط، برای قرار دادن شماره خط استفاده می‌شود. منظور از optional یعنی قرار دادن آن اختیاری می‌باشد. از شماره خط برای افزایش خوانایی برنامه استفاده می‌شود. اگر می‌خواهیم از شماره خط استفاده کنیم باید آن‌ها را در ستون اول بنویسیم. شماره خطوط در حین ترجمه در نظر گرفته نمی‌شود. توجه شود که فقط ما از این شماره خطوط استفاده می‌کنیم و GPSS برای مسائلی مثل ارجاع به یک خط، از شماره واقعی خطوط استفاده می‌کند.



شکل ۹

شماره این خط از نظر GPSS، ۴ است چون چهار سطر از ابتدای فایل به پایین آمده‌ایم.

Label : فیلد Label به ما اجازه می‌دهد تا به Entity ها یک نام بدهیم و با این نام داده شده از آن‌ها استفاده کنیم.

قوانین نامگذاری در GPSS به شرح زیر است.

تمام کاراکترهای عددی یا الفبایی و یا _ (آندرلاین) می‌توان برای نام‌گذاری استفاده کرد با این شرط که نام‌ها با حروف الفبا آغاز می‌شوند و طول هر نام به میزان ۲۰۰ کاراکتر می‌تواند باشد. همچنین نام‌ها نباید از keyword های GPSS باشند.

Verb: از Keyword های GPSS هستند. Verb ها باید نام یکی از Command ها و یا Block ها باشند. منظور از required یعنی باید نوشته شوند.

Operands: بسته به verb می‌تواند optional یا required باشد.

Comment: برای قرار دادن توضیحات استفاده می‌شود و قرار دادن آن اختیاری است.

Line number (optional) | Label (optional) | Verb (required) | Operands (depend on verb) | Comment (optional).

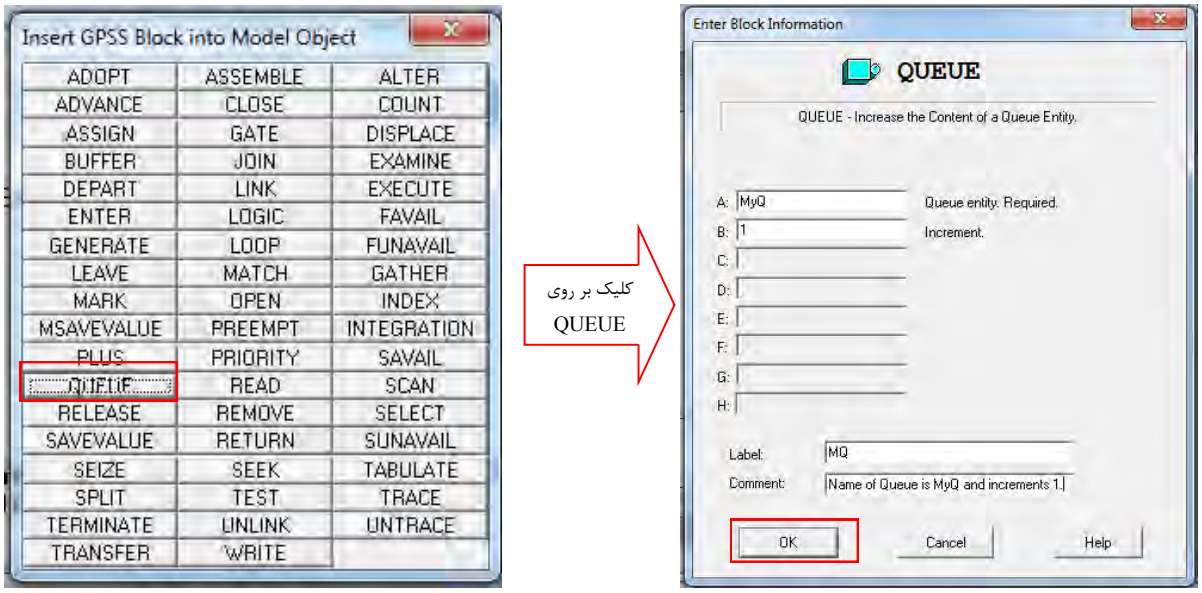
مثال: 1 MQ QUEUE MyQ,1 ; Name of Queue is MyQ and increments 1.

نحوه وارد کردن دستورات در مدل:

در مثال اوّل که با هم دیدیم شما دستورات را در Model کپی کردید! امّا اگر خواسته باشید خودتان این دستورات را بنویسید از روش‌های زیر می‌توانید استفاده کنید.

۱- تایپ کردن: کافی است تا دستورات را در صفحه مدل تایپ کنید.

۲- استفاده از Insert GPSS Block: برای وارد کردن دستورات بلوکی می‌توانید از منوی Edit بر روی Insert GPSS Block ... کلیک کنید. پس از کلیک کردن پنجره‌ای باز می‌شود که حاوی دستورات بلوکی است (شکل ۱۰). با کلیک بر روی هر کدام از دستورات، پنجره دیگری باز می‌شود که باید قسمت‌های دیگر دستور مثل Label، Operand و ... را وارد کنید. سپس بر روی دکمه OK کلیک کنید. دستور مورد نظر در مکانِ مکان‌نما (cursor) قرار می‌گیرد.

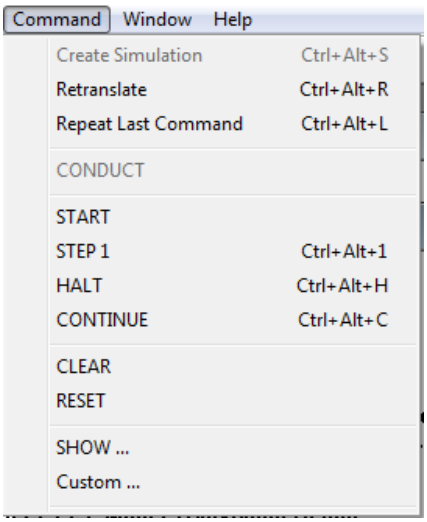


شکل ۱۰

Interactive Statement

بعد از ایجاد Simulation Object، می‌توانیم یک Model Statement را به یک Simulation Object موجود ارسال کنیم. این دستورات، Interactive Statement نامیده می‌شوند که می‌تواند حتی دستورات PLUS، Block و یا Command باشد. دستورات Interactive پس از ترجمه به Simulation Object اضافه می‌شوند و به وسیله آن‌ها می‌توان Simulation موجود را کنترل کرد (مثلاً آغاز کرد (START)، متوقف کرد (HALT) و ادامه داد (CONTINUE)).

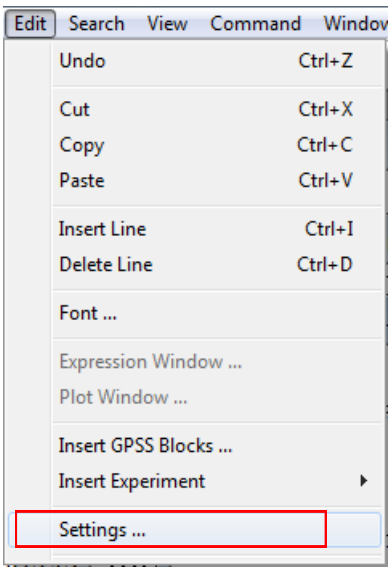
نحوه وارد کردن دستورات Interactive: ۱- در منوی Command لیستی از دستورات مهم‌تر Interactive را می‌بینید (شکل ۱۱) که می‌توانید با کلیک بر روی آن‌ها، از آن‌ها استفاده کنید.

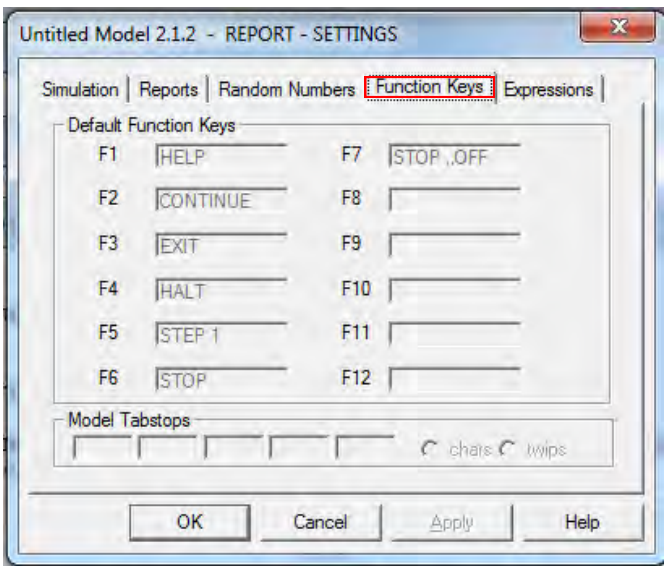


شکل ۱۱

۲- منوی Command، گزینه ... Custom. با کلیک بر روی گزینه Custom پنجره‌ای نمایش داده می‌شود که می‌توان در آن دستورات را نوشت و با کلیک بر روی، آن دستور را اجرا کرد.

۳- منوی Edit، گزینه ... Settings. با کلیک بر روی گزینه Settings پنجره‌ای باز می‌شود. در این پنجره به برگه (Tab) Function Keys می‌رویم. در این‌جا می‌توانیم یک Function Key برای دستور مورد نظر خود وارد کنیم. یعنی برای تعدادی از دستورات کلیدهایی را قرار دهیم که با فشردن آن‌ها، دستور متناظر آن‌ها اجرا شود (شکل ۱۲).





شکل ۱۲

خطایابی در Model Statement

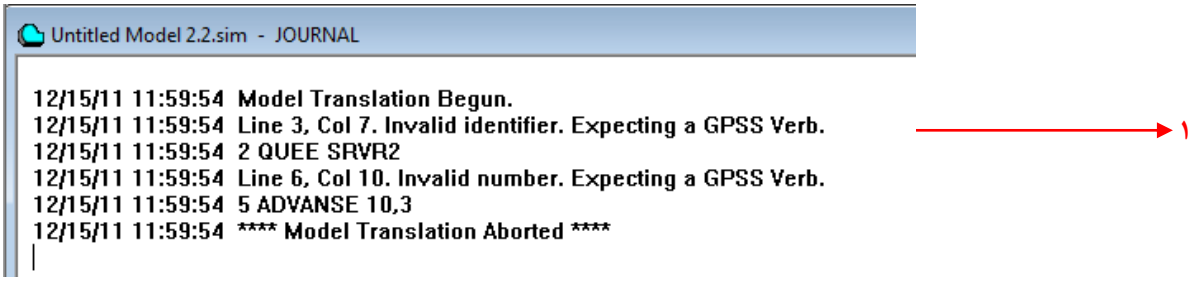
GPSS این امکان را فراهم می‌کند تا بتوان خطاهای موجود در Model Statement را شناسایی کرد. مثال زیر را انجام دهید.

مثال: نرم‌افزار GPSS را باز کنید. از منوی فایل یک مدل ایجاد کنید.

دستورات زیر را وارد نمایید.

```
*ONE SERVER ONE LINE SYSTEM
1 GENERATE 10,2
2 QUEE SRVR2
3 SEIZE SRVR1
4 DEPART SRVR2
5 ADVANSE 10,3
6 RELEASE SRVR1
7 TERMINATE 1
```

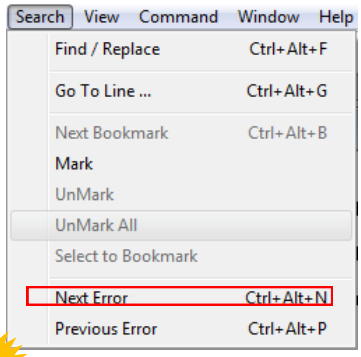
سپس از منوی Command بر روی Create Simulation کلیک کنید. پنجره باز شده مانند شکل ۱۳ می‌باشد.

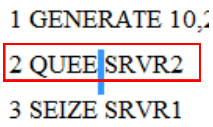


شکل ۱۳

به پیام‌های این پنجره دقت کنید. عبارت ۱ به ما می‌گوید که در سطر ۳ و ستون ۷، یک شناسه نامعتبر دیده شده، در صورتی که طبق ساختار دستور در GPSS، آن‌جا محل قرارگیری یک verb می‌باشد. کاملاً درست می‌گوید چون ما به اشتباه به جای دستور QUEUE عبارت QUEE را نوشته‌ایم (همانطور که قبلاً هم گفته شد GPSS از شماره خطهایی که ما قرار داده‌ایم استفاده نمی‌کند و برای ارجاع به شمارش تعداد سطرها از بالای فایل می‌پردازد). پیام دیگر نیز به همین ترتیب به خطای موجود در عبارت ADVANSE اشاره می‌کند (صحیح آن ADVANCE است).

راه دیگر برای یافتن خطاها استفاده از منوی Search گزینه Next Error و Previous می‌باشد. با کلیک بر روی این گزینه‌ها مکان‌نما (cursor) در کنار واژه نادرست قرار می‌گیرد.





شکل ۱۴

همانطور که در مثال ۱ دیدیم، بعد از اینکه دستور START را از منوی Command اجرا کردیم پنجره گزارش (Report) نمایش داده شد. این پنجره به طور خودکار پس از اتمام شبیه‌سازی (Simulation) به نمایش در می‌آید. در پنجره گزارش اطلاعاتی از شبیه‌سازی انجام شده در اختیارمان قرار گرفته می‌شود. اینکه چه مواردی در پنجره گزارش (Report) نمایش داده شود، قابل تنظیم است. از منوی Edit بر روی گزینه Settings ... کلیک کنید. در پنجره نمایش داده به برگه (Tab) Reports بروید. در این پنجره می‌توانید تنظیمات مختلف را انجام دهید (شکل ۱۵).

گزینه Create Standard Reports: اگر تیک این گزینه را بردارید، ساخت گزارش بعد از اتمام شبیه‌سازی به طور اتوماتیک انجام نمی‌شود.

In Windows : اگر تیک این گزینه را بردارید، به جای نمایش گزارش در پنجره، آن‌را در فایلی با پسوند gpr. ذخیره می‌کند.

Six Point : اگر فعال باشد اعداد در صفحه Report و یا در قسمت status (قسمت میانی در پایین صفحه) تا ۶ رقم اعشار (به جای ۳ رقم اعشار) نمایش داده می‌شوند.

با انتخاب کردن یا انتخاب نکردن برخی از گزینه‌ها نیز می‌توانیم معین کنیم که چه اطّلاعاتی در صفحه گزارش نمایش داده شود و یا نمایش داده نشود.

شکل ۱۵

اطّلاعات داخل پنجره گزارش: در این قسمت به توضیح اطّلاعات موجود در داخل پنجره گزارش می‌پردازیم. گزینه‌های موجود در پنجره Report مثال ۱ (به صورت default) مانند شکل صفحه بعد است.

پنجره Report مثال ۱

```
                GPSS World Simulation Report - Untitled Model 1.1.1


                     Monday, December 19, 2011 01:04:44

 START TIME            END TIME  BLOCKS  FACILITIES  STORAGES
      0.000            2537.054     7        1          0


    NAME                          VALUE
SRVR1                           10001.000
SRVR2                           10000.000


 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE           253             0       0
                    2    QUEUE              253             2       0
                    3    SEIZE              251             1       0
                    4    DEPART             250             0       0
                    5    ADVANCE            250             0       0
                    6    RELEASE            250             0       0
                    7    TERMINATE          250             0       0

FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 SRVR1             251    0.984       9.943  1      251    0     0     0      2

QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 SRVR2               4    3    253     25     0.978      9.803     10.878   0

CEC XN    PRI          M1      ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   251     0        2510.089    251      3       4

FEC XN    PRI          BDT     ASSEM  CURRENT  NEXT  PARAMETER    VALUE
   254     0        2543.722    254      0       1
```

توضیح اطّلاعات:

بخش اطّلاعات عمومی

Title:

نام مدل(Model file) یا عنوان خروجی گزارش‌گیری تولید شده با زمان و تاریخ آن را نمایش می‌دهد.

Start Time:

زمان شروع شبیه سازی را در سیستم بیان می‌کند.

End Time:

زمان سیستم را در حالتی که تعداد Transactionهای تعریف شده در سیستم شبیه‌سازی به پایان برسند را نمایش می‌دهد.

Blocks:

تعداد دستورات بلوکی موجود در کد شبیه سازی را نمایش می‌دهد.

Facilities:

تعداد Facility (سرویس دهنده) استفاده شده در شبیه سازی را نمایش می‌دهد.

Storages:

تعداد Storages (انباره‌ها) استفاده شده در شبیه سازی را نمایش می‌دهد.

بخش Block

Names:

نام‌هایی که کاربر در Simulation (شبیه‌سازی) خود استفاده می‌کند.

Values:

مقادیری که سیستم به Name های Simulation (شبیه‌سازی) نسبت می‌دهد را گویند که این مقدار از ۱۰۰۰۰ شروع می‌شود.

Loc:

شماره هر یک از بلوک‌ها در شبیه‌سازی می‌باشد.

Block Type:

بلوک‌های موجود در شبیه‌سازی را مشخص می‌کند.

Entry Count:

تعداد Transaction هایی را که در هر بلوک وارد می‌شوند (قبل از دستور Clear یا Reset) را مشخص می‌کند.

Current Count:

تعداد Transaction هایی موجود را که در هر بلوک پس از پایان شبیه سازی قرار دارد را مشخص می‌کند.

Retry:

تعداد Transaction هایی را که در پشت هر دستوربلوکی (با توجه به شرایط آن بلاک) منتظر می‌ماند را مشخص می‌کند.

بخش Facilities

Facility:

اسم یا شماره سرویس دهنده (Facility) را مشخص می‌کند.

Entries:

تعداد دفعاتی که سرویس دهنده (Facility) توسط سرویس گیرنده (Transaction) اشغال می‌شود.

AVE. TIME

میانگین زمانی که هر سرویس گیرنده (Transaction) از سرویس دهنده (Facility) استفاده می‌کند.

AVAIL.

در دسترس بودن یا نبودن سرویس دهنده را پس از شبیه‌سازی مشخص می‌کند (صفر یعنی در دسترس نبودن و یک یعنی در دسترس بودن).

OWNER.

تعداد سرویس گیرنده‌هایی که از سرویس دهنده استفاده کرده‌اند.

PEND.

تعداد سرویس گیرنده‌هایی که سرویس دهنده را با دادن وقفه به آن(سرویس دهنده)، آن را در دست گرفته‌اند.

INTER.

تعداد سرویس گیرنده هایی که هم اکنون (در زمان تهیه گزارش(شاید گزارش در وسط شبیه سازی انجام شده باشد)) از سرویس دهنده استفاده می کنند را مشخص می‌کند.

RETRY.

تعداد Transaction هایی را که در پشت این سرویس دهنده (با توجه به شرایط آن سرویس دهنده) منتظر می‌ماند را مشخص می‌کند.

DELAY.

تعداد سرویس گیرنده‌هایی که منتظر سرویس دهنده برای انجام سرویس‌دهی می‌مانند را مشخص می‌کند.

بخش Queue

QUEUE.

نام یا شماره هر صف را مشخص می‌کند.

MAX.

بیشترین مقدار طول صف را مشخص می‌کند.

CONT.

طول صف را در موقع گزارش‌گیری (شاید آخر شبیه‌سازی) را مشخص می‌کند.

ENTRY.

کل تعداد سرویس دهنده‌هایی که وارد صف شده‌اند.

Entry(0):

تعداد افرادی که بدون انتظار در صف سرویس شده‌اند.

AVE CONT:

میانگین طول صف را مشخص می‌کند.

AVE.TIME:

میانگین زمانی که هر نفر در صف قرار داشته است.

AVE.(0):

میانگین زمانی که هر نفر در صف قرار دارد؛ برای افرادی که بدون انتظار در صف سرویس شده‌اند.

RETRY:

تعداد Transaction هایی که در پشت این صف (با توجه به شرایط آن صف) منتظر می‌مانند را مشخص می‌کند.

(CURRENT EVENTS CHAIN) CEC

XN:

تعداد Transaction هایی که در زنجیره رویدادهای شبیه سازی رخ داده است را نشان می‌دهد.

PRI:

اولویت زمان‌بندی Transaction ها را نشان می‌دهد.

ASSEN:

تعداد Transaction ها را نشان می‌دهد.

CURRENT:

تعداد بلوک‌هایی که Transaction ها در آخر شبیه‌سازی در آن‌ها وجود دارند.

(FUTURE EVENT CHAIN) FEC

NEXT:

شماره بلوک زمان‌بندی شده بعدی که Transaction به آن وارد می‌شود.

PARAMETER:

اسم یا شماره پارامترهای Transaction را را نشان می‌دهد.

VALUE:

مقدار پارامتر Transaction را را نشان می‌دهد.

XN:

تعداد Transaction هایی که در زنجیره بعدی رویداد های شبیه‌سازی رخ داده است را نشان می‌دهد.

PRI:

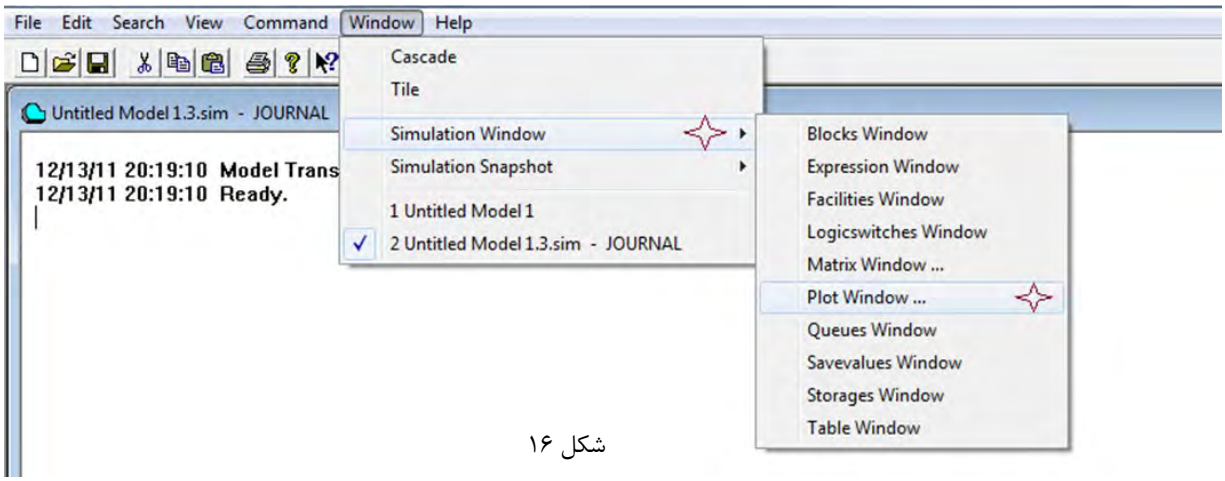
اولویت زمان‌بندی Transaction ها را نشان می‌دهد.

نمودارهای GPSS

در این بخش قصد داریم متغیّرهایی که دارای مقادیر متغیّراند را به وسیله این نرم‌افزار در نمودارهای GPSS رسم نماییم.

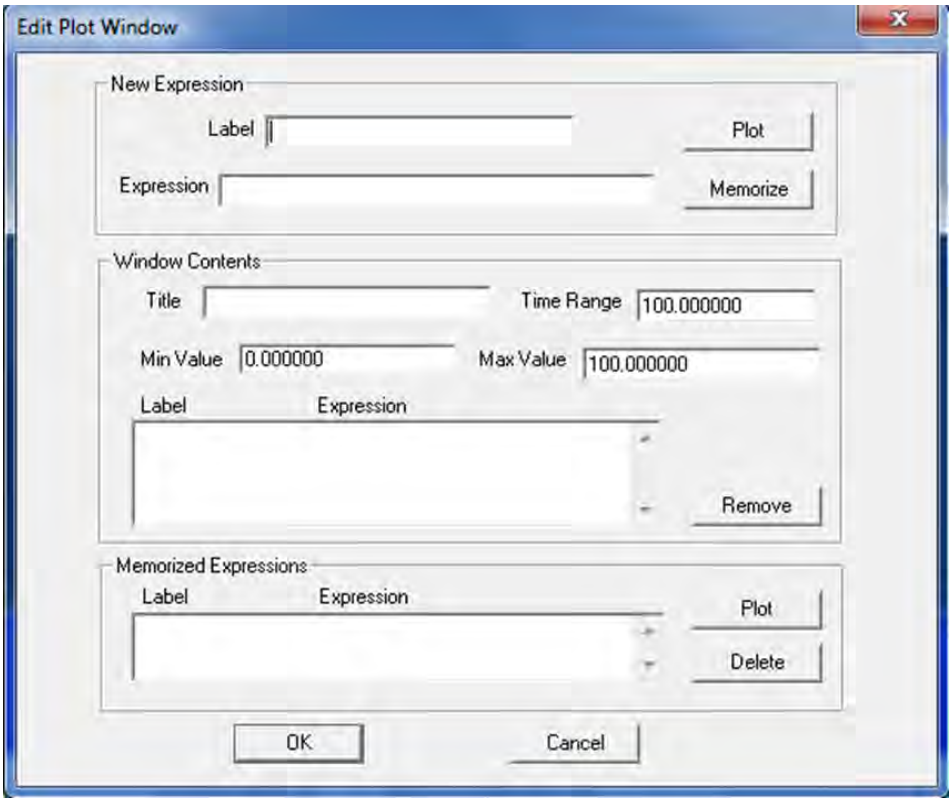
ابتدا با باز کردن یک مدل جدید Code شبیه سازی مثال ۱ را وارد کرده و بدین منظور از منوی Command گزینه Create Simulation را انتخاب کنید. توجه داشته باشید که در انتهای کد شبیه سازی خود از کلمه Start استفاده ***ننمایید***.

به این منظور متغیّر خود را تعیین نموده و سراغ پنجره‌ای که در شکل ۱۶ مشاهده می‌کنید، بروید:



شکل ۱۶

حال پنجره‌ای به شکل ۱۷ ظاهر خواهد شد:



شکل ۱۷

در این پنجره نامی دلخواه برای متغیری که در نظر گرفته بوده‌اید، در قسمت Label وارد نمایید.

با توجه به اینکه نمودار می‌بایست از نوع متغیّر شما آگاهی پیدا کند، باید در قسمت Expression از فرمت زیر استفاده نمایید.

FORMAT$OBJECT

در این فرمت به جای OBJECT اسم متغیّر انتخاب شده را به طور دقیق قرار داده و با توجه به جدول زیر FORMAT را نیز جایگزین نمایید.

FORMAT	Proto Type
Q	QUEUE
QA	Average Queue
S	Storage in use
W	Current Block count
١TG	Remaining Termination Count

به طور مثال می‌توان در Expression نوشت : Q$srvr2.

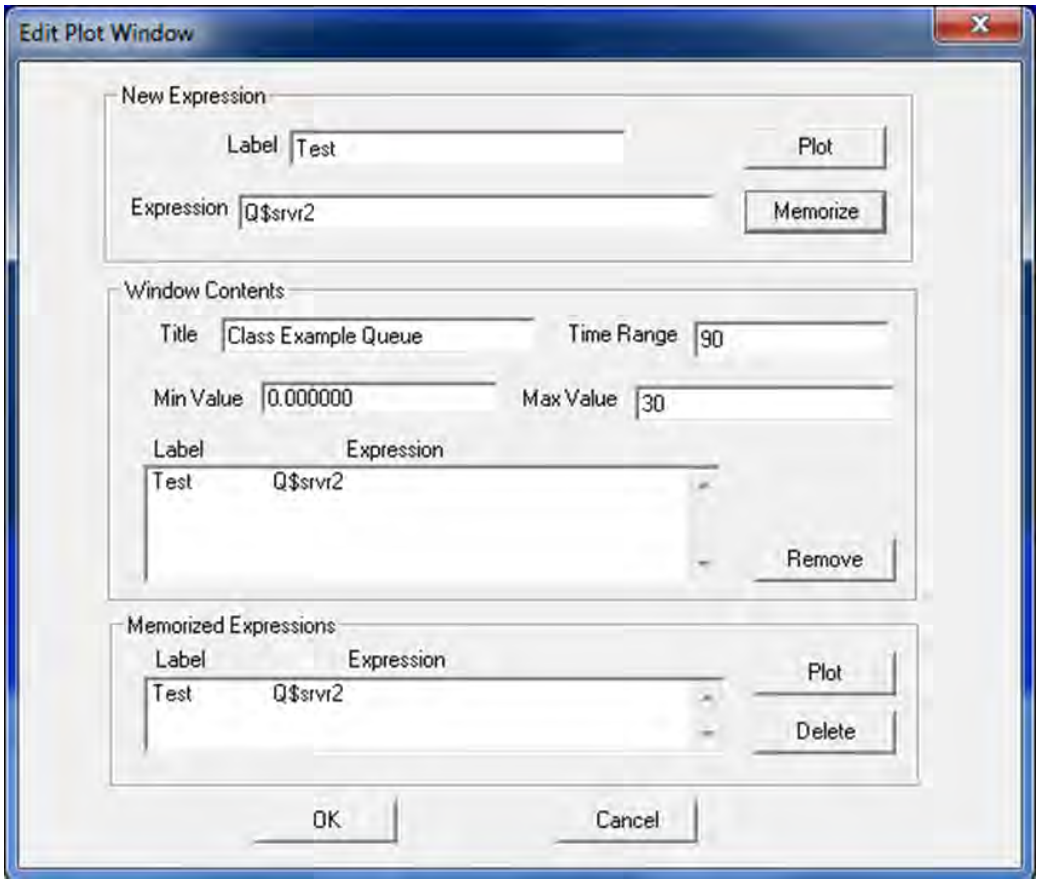
این مدل فرمت ذکر شده، جزء معدود فرمت‌های با قاعده است که ممکن است مورد نیاز واقع شود، لذا فرمت‌های موجود در شبیه‌سازی محدود به فرمت مذکور نیست. مثل X_ که برای رسم تابع Sin از آن استفاده می‌شود.

بعد از پر کردن این دو فیلد دو دکمه Plot و Memorize را فشار بزنید تا به نمودار اضافه شود.

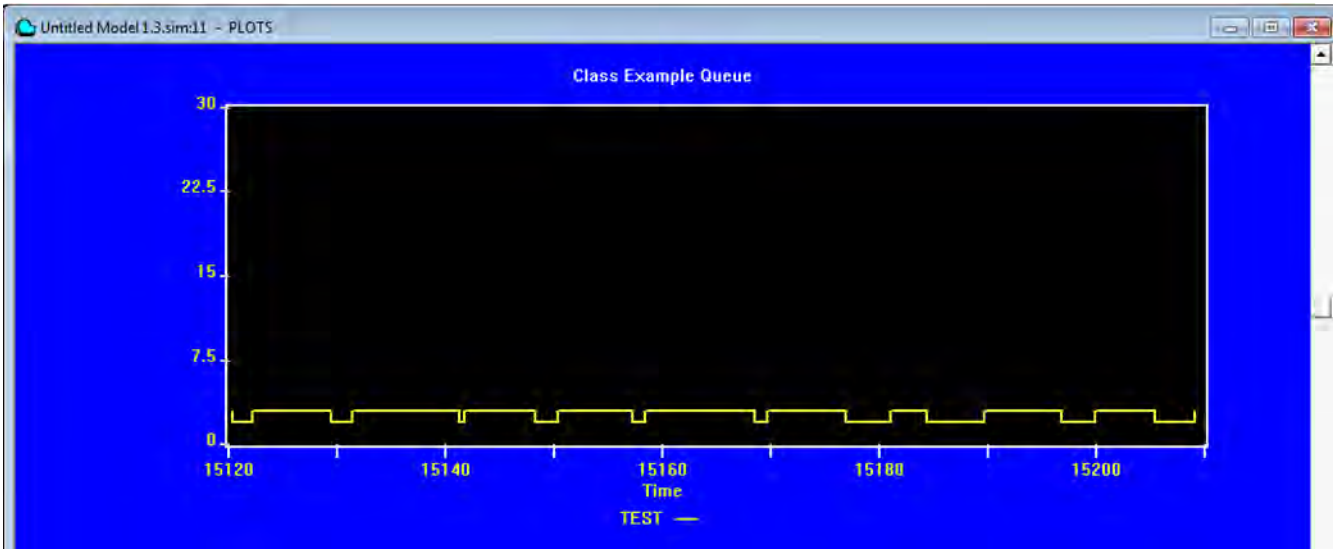
در گام بعدی بهتر است Time Range و MaxValue را از حالت Default به حالت‌های مناسب‌تری در بیاورید تا نمودار بهتری را مشاهده نمایید. (در شکل ۱۸ یک نمونه کامل شده از آن را خواهیم دید.)

ضمناً اگر بخواهید بعد از اجرا شدن، دوباره از اطّلاعات پر شده قبلی استفاده کنید، در صورت وجود رکوردی در Memorized Expression آن را دوباره Plot کنید.

سپس در انتها کلید OK را زده و از منوی بالا Command\START را انتخاب کنید. در مقابل شما Textbox ای ظاهر می‌شود که در آن Start 1 نوشته شده است. عدد یک نشان دهنده دوره تکرار است، آن را به طور مثال به ۱۰ تغییر دهید تا نموداری همانند نمودار شکل ۱۹ مشاهده می‌نمایید.



شکل ۱۸



شکل ۱۹

برخی از Command ها

در بخش‌های قبلی با Command ها آشنا شدیم. در اینجا برخی از این Command ها را شرح می‌دهیم.

یادآوری:

Model Statement
- GPSS Statement
  - Block Statement
  - Command Statement
- PLUS Procedure

⟹ *VARIABLE*

ساختار:

Var VARIABLE X

مثال:

Var1 VARIABLE 5#LOG(Q$WaitingLine)

⟹ *FVARIABLE*:

ساختار:

NAME FVARIABLE X

مثال:

VAR1 FVARIABLE 5#LOG(Q$WAITINGLINE)

یک متغیّر name ایجاد، مقدار x را یک مقدار اعشاری نسبت می‌دهد.

⟹ *EQU* :

ساختار:

NAME EQU X

مثال:

Price EQU 19.95

یک متغیّر به نام name ایجادکرده و به آن مقدار x را اختصاص می‌دهد.

⟹ *INCLUDE:*

ساختار:

INCLUDE "FileName.TXT"

مثال:

INCLUDE "SAMPLE1.TXT"

می‌توان دستورات شبیه سازی را که درفایل متنی وجود دارد به وسیله دستورات بالا به فایل شبیه سازی اضافه کرد.

(فایل متنی باید درمسیری که فایل شبیه سازی ذخیره شده است قرار داشته باشد.)

⟹ *INITIAL*

ساختار:

INITIAL A,B

برای مقدار اولیه دادن به متغیّر هست که دراینجا به متغیّر A مقدار B رامی‌دهد.

مثال:

INITIAL MX$Inventory(Part_39,Stocklevel),200

در اینجا MXمشخص می‌کندکه متغیّر از نوع Matrix هست، Part_39 شماره سطر و Stocklevel شماره ستون را مشخص می‌کند.

INITIAL MainResult , UNSPECIFIED

به تمام خانه‌های ماتریس MainResult مقدار UNSPECIFIED اختصاص می‌دهد.

⟹ *MATRIX:*

ساختار:

NAME MATRIX RowCount,ColumnCount

مثال:

Inventory MATRIX ,100,5

این دستور یک ماتریس با صد سطر و پنج ستون ایجاد کرده و اسم آن را Inventory قرار می‌دهد. حداکثرمی توان ماتریس ۶ بعدی تعریف کرد.

⟹ *REPORT :*

ساختار:

REPORT A,B

مثال:

REPORT

این دستور یک گزارش ایجاد می‌کندکه متغیّرA همواره باید مقدار Null داشته باشد و متغیّر B می‌تواند مقدارNull یا Now داشته باشد.

⟹ *HALT:*

ساختار:

HALT

یک وقفه در شبیه سازی ایجاد کرده و باعث پاک شدن صف شبیه سازی می‌شود.

⟹ *RESET*

ساختار:

RESET

این دستور باعث تکرار شبیه سازی می‌شود. امّا همانند دستور Clear نیست چون Transactions(سرویس گیرنده هـا) را از شبیه سازی حذف نمی‌کند.

⟹ *STEP*

ساختار:

STEP A

مثال:

STEP 1

شبیه سازی را به اندازه A بلوک اجرا می‌کند و پس از اجرای A بلوکhalt می‌کند.

⟹ *STOP*

ساختار:

STOP A,B,C

مثال:

STOP 100,52

وقتی که سرویس گیرنده A به بلوک B می‌رسد متوقف می‌شود و متغیّر C می‌تواند Null باشد.

⟹ *SHOW*

ساختار:

SHOW X

مثال:

SHOW 2#LOG(Q$Barber)

این دستور عبارت X را (که می‌تواند یک عبارت ریاضی نیز باشد) رادر Status Line نمایش می‌دهد.

⟹ *EXIT*:

ساختار:

EXIT

آن دوره از شبیه‌سازی را فورا به پایان می‌رساند.

دستورات بلوکی (Block Statement)

دستورات بلوکی از قبیل ADVANCE, LEAVE, ENTER, QUEUE, SEIZE, DEPART, ... که قبلاً با آن‌ها آشنا شده اید. در این‌جا نیز برخی از آن‌ها را آورده ایم.

یادآوری:

- Model Statement
  - GPSS Statement
    - Block Statement
    - Command Statement
  - PLUS Procedure

⟹ *OPEN*

ساختار:

OPEN ("FileName.txt"),countIden, Label

مثال:

OPEN ("Input.txt"),1,Flag1

یک فایل خواندنی، نوشتنی ایجاد می‌کند و شماره یک را برای شناسایی آن می‌دهد اگر در هنگام باز کردن به مشکل برخـورد به برچسب Flag1 بپرد.

⟹ *READ*

ساختار:

READ Var,countIden,Lable

مثال:

READ Excel,1,Finis

یک خط از فایل می‌خواند و اگر در هنگام باز کردن به مشکل برخورد به برچسب Finis بپرد.

⟹ *SAVEVALUE*

ساختار:

SAVEVALUE Var1,Var2

مثال:

SAVEVALUE Rownum+,1

به متغبر Rownum یکی اضافه می‌کند.

⟹ *MSAVEVALUE*

ساختار:

MSAVEVALUE MatrixName, Rownum,Colnum,Value

مثال:

MSAVEVALUE MatrixName1,X$Rownum,X$Colnum,Value

مقدار Valueرا در خانه ای از ماتریس که مشخص شده است قرار می‌دهد.

⟻ *TEST*

ساختار:

TEST a Val1,Val2,Label

اینگونه عمل می‌کند که شرطی راکه باتوجه به حرف a اعمال می‌شود را چک کرده اگر درست بود اجازه اجرای برنامه را می‌دهد و اگرنه به مکان Label می‌پرد. حرف a که در بالا آمده است می‌تواند یکی از حروف زیر باشد.

E دوتا مقدار باید با هم مساوی باشند.

G مقدار اوّلی از دومی باید بزگتر باشد.

GE مقدار اوّلی از دومی باید بزگتر یا مساوی باشد.

L مقدار اوّلی از دومی باید کوچکتر باشد.

LE مقدار اوّلی از دومی باید کوچکتر یا مساوی باشد.

NE دوتا مقدار باید با هم مساوی نباشند.

مثال:

TEST G Colnum,2,Label1

اگر عدد Colnum از شماره ۲ بیشتر بود روند را ادامه بده و گرنه به برچسب بالا بپرد.

⟹ *WRITE*

ساختار:

WRITE "STRING",countIden,Label

مثال:

WRITE "TEST",2,lb3

مقدار رشته را در فایل با شناسه ۲ می‌نویسد و اگر به مشکل برخورد کرد به برچسب بالا بپرد.

⟹ *TRANSFER*

ساختار:

TRANSFER , Label

مثال:

TRANSFER ,lb3

کنترل اجرای برنامه را به برچسب lb3 منتقل می‌کند.

⟹ *CLOSE*

ساختار:

CLOSE CountProb,countIden,Lable

مثال:

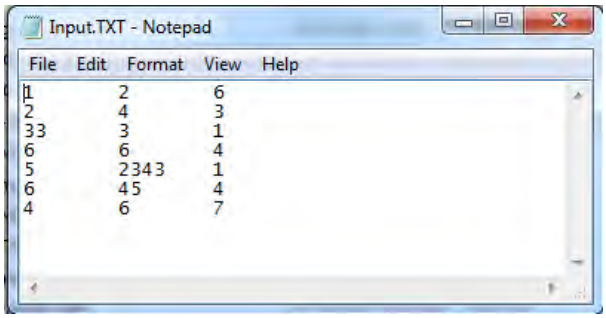
CLOSE Prob2,2,Flag3

برنامه ای را که با شماره ۲ مشخص کرده است را می‌بندد و اگر در حین بسته شدن فایل به مشکل برخورد کند به برچسب Flag3 بپرد وشماره Error آن را در متغیّر Prob2 قرار دهد.

***مثال**: برای درک بهتر دستورات بلوکی، مثالی را در اینجا آورده‌ایم. در این مثال، اطّلاعات از فایل با نام Input.txt خوانده شده و در فایلی به نام Output.txt ریخته می‌شود (چاپ می‌شود).

پاسخ: برنامه کامل این مثال در صفحات ۲۸ و ۲۹ آورده شده است.

آن را Copy کنید و در یک Model File (File -> New/ Model)، Paste نمایید. همانطور که در سوال گفته شده است، برنامه باید اطّلاعات را از یک فایل با نام Input.txt بخواند. محتویات فایل Input.txt مانند شکل ۲۰ است. لطفاً قبل از اجرای برنامه یک فایل متنی (.txt) با نام Input، در مسیری که Model File تان را ایجاد کرده‌اید، ایجاد کنید و اطّلاعات زیر را در آن کپی کنید (شکل ۲۰) (توجه کنید که نیازی به ایجاد فایل Output.txt نیست، این فایل پس از اجرای برنامه به طور خودکار ایجاد می‌شود).

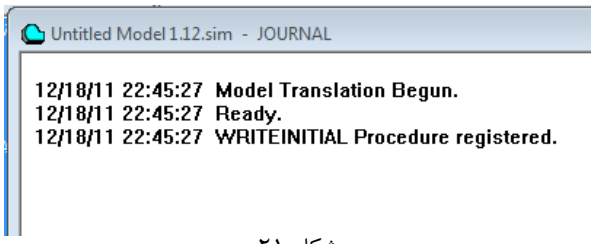


شکل ۲۰

1	2	6
2	4	3
33	3	1
6	6	4
5	2343	1
6	45	4
4	6	7

این اطّلاعات را با همین فرمت در فایل Input.txt کپی کنید.

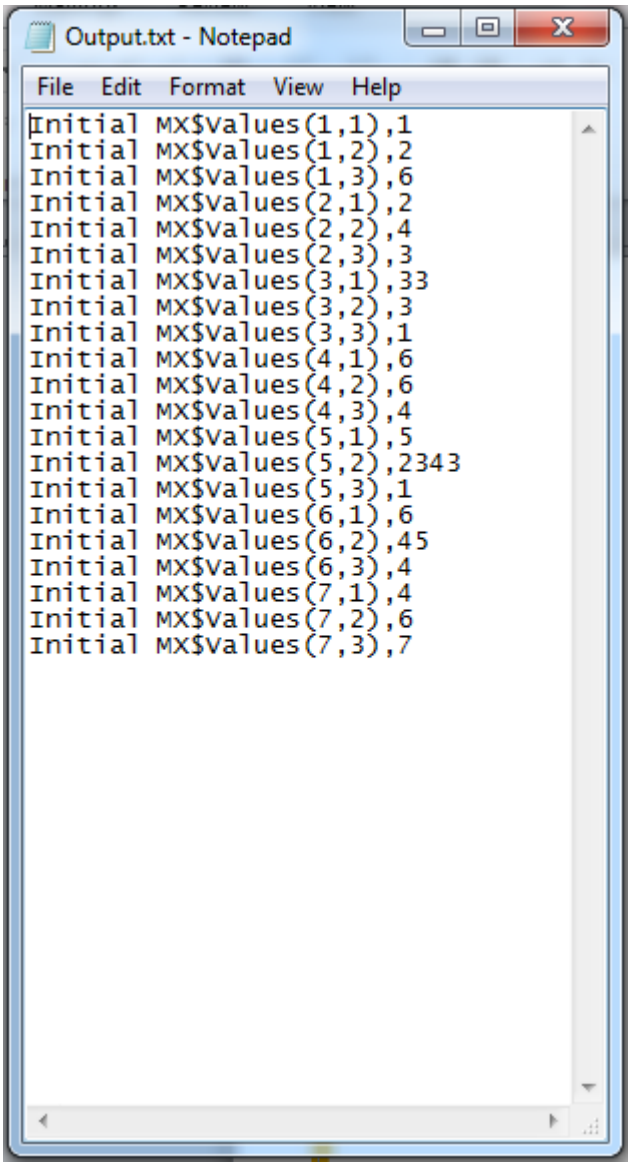
حالا می‌توانید برنامه را اجرا کنید. از منوی Command، گزینه Create Simulation را انتخاب کنید، پس از انتخاب Create Simulation، پنجره‌ای مانند شکل ۲۱ را مشاهده می‌کنید.



شکل ۲۱

سپس از گزینه Command گزینه START را انتخاب و در پنجره باز شده بر روی OK کلیک کنید.

حال به مسیر ذخیره Model File بروید، فایلی با نام Output.txt را خواهید دید که حاوی اطّلاعات فایل Input.txt است (البته با فرمتی دیگر که در برنامه تعریف شده است) (شکل ۲۲).



شکل ۲۲

شروع کد کامل برنامه

```
Values  MATRIX      ,7,3 ;Defines Matrix.
        GENERATE    ,,,1 ;Produce only one Xact.
        OPEN ("Input.txt"),1,Flag ;Open external file to be read.
        OPEN ("Output.txt"),2,Flag1 ;Open output file to which output
 ;                                  will be written.
Again   READ Excel,1,Finis ;Read Inpt.dat one line
 ;                                  at a time into parameter.
 ;                                  Excel. At end or error;
 ;                                  go to label Finis.
        SAVEVALUE Rownum+,1      ;Increase Matrix row by 1.
Morecol SAVEVALUE Colnum+,1      ;Increase Matrix column by 1.
        MSAVEVALUE Values,X$Rownum,X$Colnum,(Word(P$Excel,X$Colnum))
                                 ;Store the value in the parameter
 ;                                  in the Matrix cell.
        TEST G    X$Colnum,2,Morecol ;Are there more columns to be
loaded?
        SAVEVALUE Colnum,0       ;No more columns? Start next row
        TRANSFER  ,Again         ; at label Again.
Finis   CLOSE     Prob,1,Flag2   ;Close data file.
        ASSIGN    Colnum,0       ;Prepare to cycle
        ASSIGN    Rownum,0       ; thru Values Matrix
Again2  ASSIGN    Rownum+,1      ; from the beginning
More    ASSIGN    Colnum+,1
         TEST LE   P$Rownum,7,Finis2 ;Has final row been processed?
         WRITE (WriteInitial()),2,Wrtebug ;Write Initial
 ;                                  Statement using
 ;                                  PLUS Procedure.
         TEST G    P$Colnum,2,More ;Are there more columns?
         ASSIGN    Colnum,0      ;Set Column number to 0.
           TRANSFER  ,Again2       ;Start Next row.
```

```
Finish2  CLOSE     File2,2,Flag3 ;Close output file.
TERMINATE 1              ;End for normal completion.
Flag    TERMINATE 1              ;End if errors in input file Open.
Flag1   TERMINATE 1              ;End if errors in output file
 ;                                 Open.
Flag2   TERMINATE 1              ;End if errors in input file
 ;                                 Close.
Wrtebug TERMINATE 1              ;End if errors on Write.
Flag3   TERMINATE 1              ;End if errors on output file
 ;                                 Close
PROCEDURE WriteInitial() BEGIN
    TEMPORARY Value1,Value3,Value5,Value6,Value7;
    Value1="Initial MX$Values(";
    Value3=",";
    Value5="),";
    Value6=MX$values(P$Rownum,P$Colnum);
    Value7=" ";
    RETURN (PolyCatenate
        (Value1, P$Rownum, Value3,P$Colnum,Value5,Value6,Value7));
END;
```

پایان کد برنامه

حال به توضیح کد مثال قبل می‌پردازیم.

```
Values  MATRIX      ,7,3 ;Defines Matrix.
        GENERATE    ,,,1 ;Produce only one Xact.
        OPEN ("Input.txt"),1,Flag ;
```

یک فایل خواندنی نوشتنی ایجاد می کند و شماره یک را برای شناسایی آن می‌دهد اگر در هنگام باز کردن به مشکل برخورد به برچسب فلگ۱ بپرد.

```
        OPEN ("Output.txt"),2,Flag1 ;
```

یک فایل خواندنی نوشتنی ایجاد می‌کند و شماره دو را برای شناسایی آن می‌دهد و اگر در هنگام باز کردن به مشکل برخورد به برچسب فلگ۲ بپرد.

```
Again   READ Excel,1,Finis ;
```

یک خط از فایل فقط خواندنی می‌خواند و اگر در هنگام باز کردن به مشکل برخورد به برچسب فینیس بپرد.

```
        SAVEVALUE Rownum+,1       ;سطر ماتریس را یکی زیاد میکند
Morecol SAVEVALUE Colnum+,1       ;ستون ماتریس را یکی زیاد میکند
        MSAVEVALUE Values,X$Rownum,X$Colnum,(Word(P$Excel,X$Colnum));
```

از متغیّر اکسل (که حاوی یک سطر از فایل می‌باشد) به اندازه شماره ستون جلو می‌رود ویک کلمه می‌خواند ودر آن خانه از ماتریس قرار می‌دهد.

```
        TEST G    X$Colnum,2,Morecol ;
```

اگر عدد حداکثر تعداد ستون(۲)از شماره ستون بیشتر بود روند را ادامه بده وگرنه به برچسب مقابل بپرد.

```
        SAVEVALUE Colnum,0 ;
```

چون ستون‌های یک سطر همه خوانده شده‌اند پس برای سطر بعد باید تعداد ستون‌ها صفر شود

```
        TRANSFER  ,Again          ;  به برچسب مقابل بپر (برای خواندن سطر بعد)
Finis   CLOSE     Prob,1,Flag2    ;
```

فایل فقط خواندنی را ببند و اگر در حین انجام آن به مشکل برخورد به برچسب فلگ۲ بپرد و کد پیغام خطا را در متغییر مقابل قرار می‌دهد.

از اینجا به بعد فایل را باید نوشت

```
        ASSIGN    Colnum,0
        ASSIGN    Rownum,0
Again2  ASSIGN    Rownum+,1   ;     نوشتن فایل را از سطر و ستون یک شروع میکنیم
More    ASSIGN    Colnum+,1
        TEST LE   P$Rownum,7,Finis2 ;
```

اگر تعداد سطر ها کمتر یا مساوی آخرین سطر(۷) بود روند را ادامه بده والا به برچسب فینیس بپر.

```
        WRITE (WriteInitial()),2,Wrtebug ;
```

مقدار بازگشتی از تابع را در فایل می نویسد واگر به مشکل برخورد کرد به برچسب مقابل بپرد.

```
        TEST G    P$Colnum,2,More ;
```

اگر عدد حداکثر تعداد ستون (۲) از شماره ستون بیشتر بود به برچسب مقابل بپرد

```
        ASSIGN    Colnum,0        ;
```

چون ستون‌های یک سطر همه نوشته شده‌اند پس برای سطر بعد باید تعداد ستون‌ها صفر شود

```
        TRANSFER  ,Again2         ;به برچسب مقابل بپر (برای خواندن سطر بعد)
Finis2  CLOSE     Prob2,2,Flag3   ;فایل فقط نوشتنی را ببند
        TERMINATE 1               ;شبیه سازی در حالت نرمال به پایان می رسد
Flag    TERMINATE 1               ;
```

شبیه سازی به پایان می‌رسد در حالتی که فایل فقط خواندنی به دلیل مشکل باز نشده است

```
Flag1    TERMINATE 1                ;
```
شبیه‌سازی به پایان می رسد در حالتی که فایل فقط نوشتنی به دلیل مشکل باز نشده است
```
Flag2    TERMINATE 1                ;
```
شبیه‌سازی به پایان می رسد در حالتی که فایل فقط خواندنی به دلیل مشکل بسته نشده است
```
Wrtebug  TERMINATE 1                ;
```
شبیه‌سازی به پایان می رسد در حالتی که فایل فقط نوشتنی به دلیل مشکل امّاده برای نوشتن نبوده است
```
Flag3    TERMINATE 1                ;
```
شبیه‌سازی به پایان می رسد در حالتی که فایل فقط نوشتنی به دلیل مشکل بسته نشده است

```
PROCEDURE WriteInitial() BEGIN
    TEMPORARY Value1,Value3,Value5,Value6,Value7;
    Value1="Initial MX$Values(";
    Value3=",";
    Value5="),";
    Value6=MX$values(P$Rownum,P$Colnum);
    Value7=" ";

    RETURN (PolyCatenate
```
این تابع(توسط خود نرم افزار تعریف شده است) برای الحاق چند رشته به کار می‌رود.
```
        (Value1,
        P$Rownum,
        Value3,
        P$Colnum,
        Value5,
        Value6,
        Value7));
END;
```

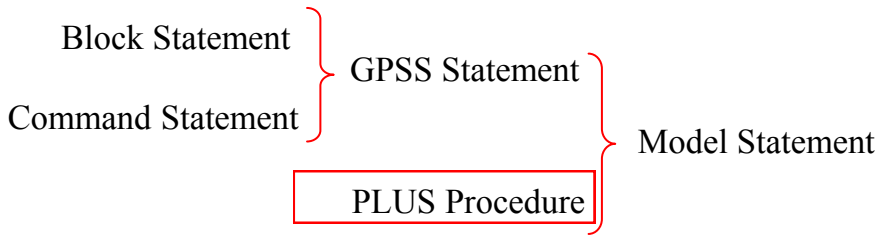
PLUS Procedure

نرم افزار GPSS علاوه بر GPSS Statement، توانایی برنامه نویسی با دستورات PLUS را نیز دارد.

PLUS: The Programming Language Under Simulation

که با توجه به آن نوع دیگری از دستورات به نام PLUS Procedures تعریف می‌شود که برخی از آن‌ها در زیر آمده است.

یادآوری:



⟹ *GOTO*

ساختار:

GOTO Label

با این دستور روند اجرای برنامه را به مکانی که توسط Label مشخص شده است منتقل می‌شود.

⟹ *IF*

ساختار:

IF (Expression) THEN Statement;

نوع دیگری از آن به صورت زیر است.

IF (Expression) THEN Statement1 ELSE Statement2

باید توجه داشت که اگر تعداد دستورات بیشتر از یک خط باشد باید آن‌ها را بلوکه کرد با توجه به مثال زیر:

IF (Expression) THEN BEGIN

Statement1

Statement2

END

⟹ *PROCEDURE*

ساختار:

PROCEDURE Name (ArgumentList) Statemen

مثال:

PROCEDURE Factorial(Arg1)

یک تابع به نام Factorial ایجاد کرده و یک پارامتر به آن ارسال می‌کند. لازم به ذکر است که برای بلوکه کردن آن از توضیح داده شده در بالا می‌توان استفاده کرد.

⟹ *WHILE*

ساختار:

WHILE(Condition) DO

مثال:

WHILE(Arg1>1) DO

تا زمانی که شرط داخل پرانتز برقرار باشد این حلقه را ادامه می‌دهد. در اینجا نیز که برای بلوکه کردن آن از توضیح داده شده در بالا می‌توان استفاده کرد.

⟹ *RETURN*

ساختار:

RETURN Arg

مثال:

RETURN Result

برای بازگشت جواب از تابع به کار می‌رود.

⟹ *TEMPORARY*

ساختار:

TEMPORARY Arg

مثال:

TEMPORARY Result

یک متغیّر موقت ایجاد می‌کند که فقط در خود برنامه استفاده می‌شود.

*در اینجا یک مثال از نوشتن تابع می‌بینیم (محاسبه فاکتوریل عدد):

```
PROCEDURE Factorial(Arg1) BEGIN
  TEMPORARY Result
  Result=1
  WHILE(Arg1>1) DO BEGIN
    Result=Result#Arg1
    Arg1=(Arg-1)
  END
  RETURN Result
END
```

این تابع را نیز به صورت بازگشتی در زیر آورده ایم (در اینجا # مانند علامت ضرب عمل می‌کند).

```
PROCEDURE Factorial(Arg1) BEGIN
  IF (Arg1<=1) THEN RETURN 1
  ELSE RETURN (Arg1#(Factorial(Arg1-1)))
END
```

منابع:

۱- جزوه درس شبیه‌سازی جناب آقای محمدیان سمنانی

۲- Reference Manual و Tutorial Manual، برگرفته از نرم‌افزار GPSS World Student Version 5.2.2