

Programming languages: a quick comparison:

**ADA:** This is a high level language developed in the late '70s for the Department of Defense. Named for Ada Lovelace, the first female programmer. ADA is intended for embedded systems (computer systems preprogrammed for a specific task and 'embedded' in the equipment they serve). Bears a strong resemblance to Pascal; stresses modularity (i.e. programs are written as a series of independent modules or routines) and maintainability (i.e. ease with which failures, glitches can be isolated and fixed). A sample of code:

```
with SQR, SIMPLE_IO;
procedure PRINT_ROOT is
  use SIMPLE_IO;
  X: FLOAT;
begin
  GET (X);
  PUT (SQR(X));
end PRINT_ROOT;
```

**APL:** Abbreviation for A Programming Language, a mathematically structured programming language popular in the '70s for problem-solving applications. In its simplest mode of operation, APL performs the functions of a programmable calculator. Its high-powered, concise code includes extended single operators that allow the user to perform operations like taking the inverse of a matrix or solving a set of linear equations. A sample of code:

```
↑ INVEST
[1] P<- 1000      (these symbols don't accurately reflect the special
APL character set)
[2] I<-5
[3] N<-10
[4] A<-PX(1+I)/100)*N
[5] A
[6] €€↑
      INVEST
1628.894627
```

**BASIC:** An acronym for Beginner's All-purpose Symbolic Instruction Code. An easy-to-learn, easy-to-use high level algebraic programming language with generally simple statement formats. Probably the most widely used language in the world. This language has been implemented on virtually every microcomputer, mini- and mainframe. Examples include TrueBASIC (from BASIC creators John Kemeny and Tom Kurtz at Dartmouth University), QuickBASIC, its precursor, MS-BASIC, and finally Visual Basic (from Microsoft Corp.) and VIP-BASIC (Mainstay Corp., for the Macintosh).. Here is a piece of a Visual BASIC program:

```
WHILE x$<>"Q"
  i=0:'reset the timer counter
  char$=CHR$(ASC("a")+INT(RND(1)*26))
  LOCATE 10,40:PRINT char$;" ";
  tymit:
  x$=INKEY$:i=i+1
  S$=S$+x$
  IF x$="" THEN
    GOTO tymit
  END IF
  IF x$="Q" THEN
    GOTO quit
  END IF
  keystrokes=keystrokes+1
  IF x$<>char$ THEN
    errors=errors+1:PRINT CHR$(7)
    GOTO tymit
  END IF
```

**C:** C was written as a structured language. A structured language has various modules and segments that can be written and tested separately and then combined to form a working application. C was also written originally for the UNIX OS. C has become the main programming language for system engineers and software designers world wide. C has some restrictions (poor type checking, unsuitable for run time manipulations)... A sample of code:

```

main()
[
    int length, width area, circum;
    length=40;
    width=30;
    area=length*width;
    circum=2*(length+width);
    printf('Length   %5d\n',length);
    printf('Width    %5d\n',width);
    printf('\n');
    printf('Area      %5d\n',area);
    printf('Circumference %5d\n',circum);
]

```

**C++:** While C and C++ are two distinct languages, the distinction between the languages has become blurred in recent years.. C++ can be used to write structured programming and the additions it made to C in type checking and uses of references, greatly add to C's power. C++ also has added a much improved method of adding comments to the code. C++ power and its wide spread use is in writing Object Oriented Programming (OOP). C++ is based on C, and it is an improved C. Most modern C/C++ compilers integrate both C and C++ into one basic compiler-- allowing you to use C++ additions when writing a structured language. Popular versions include Turbo C++, Symantec C++, Borland C++ and Visual C++ (from Microsoft); GNU C/C++ is for UNIX system machines. Microsoft Visual C++ is integrated with a built in code writing tool for Windows. It has an excellent editor and debugger. The interface is more the standard method with windows inside of a window and options pulled down from a tool bar. Code looks pretty much like that for C.

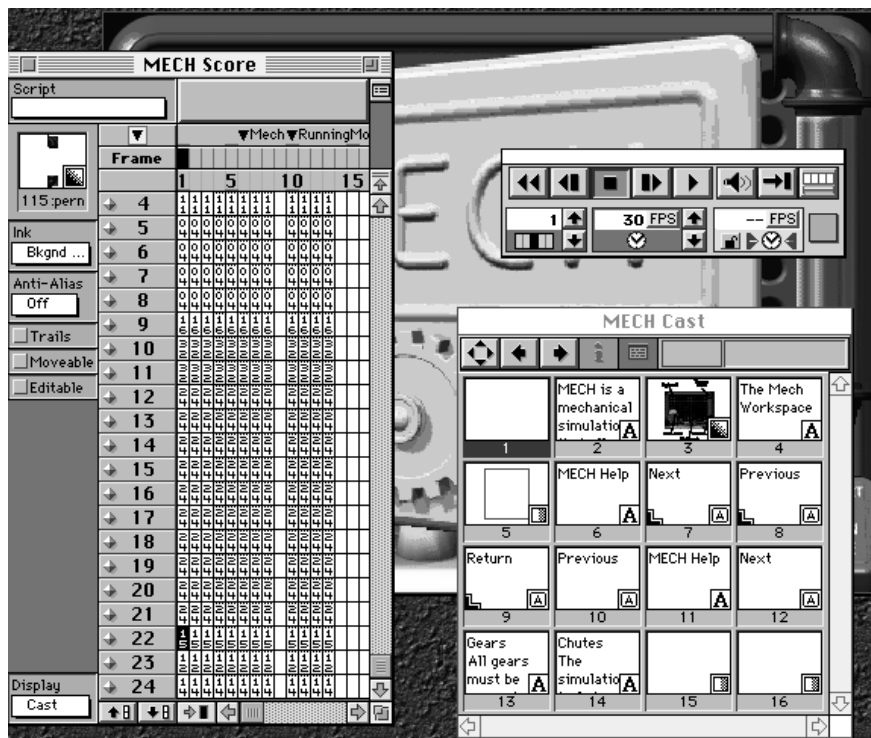
**COBOL:** An acronym for Common Business Oriented Language, a high level language developed for business data processing applications. Every COBOL source program has four divisions: (1) Identification Division identifies the source program and output of a compilation; (2) Environment Division specifies those aspects of a data processing problem that are dependent upon the physical characteristics of a particular computer. (3) Data division describes the data that the object program is to accept as input, manipulate, create or produce as output; and (4) Procedure Division specifies the procedures to be performed by the object program, using English-like statements. A sample of code:

```

00510 PROCEDURE DIVISION.
00520 START-IT.
00530     OPEN INPUT IN-FILE.
00540     OPEN OUTPUT OUT-FILE.
00550 DO-IT.
00560     MOVE SPACES TO CHECK-REGISTER.
00570     READ IN-FILE AT END GO TO END-IT.
00580     MOVE AMOUNT TO AMOUNT-OUT.
00590     MOVE CHK-NUMBER TO CHK-NUMBER-OUT
00600     MOVE ISSUED-TO TO ISSUED-TO-OUT.
00610     MOVE MONTH TO MONTH-OUT.
00630     MOVE YEAR TO YEAR-OUT.
00640     WRITE CHECK-REGISTER.
00650 END-IT.
00660     CLOSE IN-FILE, OUT-FILE
00670     STOP RUN.

```

**Director:** This software package provides for the creation and distribution of multimedia products that include 2 and 3-D graphics, text, sound, animation and digital video, synchronized in a "score." Director is also object-oriented, and the behavior and attributes of objects can be programmed in the "Lingo" scripting language. Below, an example of the four major windows of Director (score, cast, control panel, stage), and a Lingo script.



```

on startmovie
  global pausemax
  set pausemax to 3000
  set the soundEnabled = TRUE
  if the colorDepth <> 8 then
    set the colorDepth to 8
  end if
  initindicator
  resetMovie
end startmovie

```

**FORTH:** An efficient low-level programming language for use in functional control software, e.g., for instrument control. Invented by Charles Moore in the early '70s (for astronomical instrument control). Language is structured (rigorous, disciplined rules for using a few coding structures; main functions consist of low-level components organized in top-down fashion). A sample of code:

```

: FIBONACCI
1 DUP DUP DUP          (SET UP INIT VALUES)
CR . .                (PRINT FIRST 2 VALUES)
11 1                   (LOOP 10 TIMES)
DO
  DUP ROT +           (COMPUTE NEXT ELEMENT)
  DUP .               (PRINT IT)
LOOP
:OK
FIBONACCI

```

**FORTRAN:** An acronym for FORMula TRANslator. Historically the most commonly used high-level programming language for mathematical, scientific and engineering computations. Early versions included FORTRAN II and FORTRAN IV; Fortran 77 commonly used. A sample of code:

```

C AIRPLANE DISTANCE COMPUTATION
S=1.0
A=3.0
I=0
10 I=I+1
T=1
D=SQRT(A**2+(S*T)**2)
WRITE (6,20) T,D
20 FORMAT (F5.0,F10.3)
IF (I .LT. 60) GOTO 10
STOP

```

END

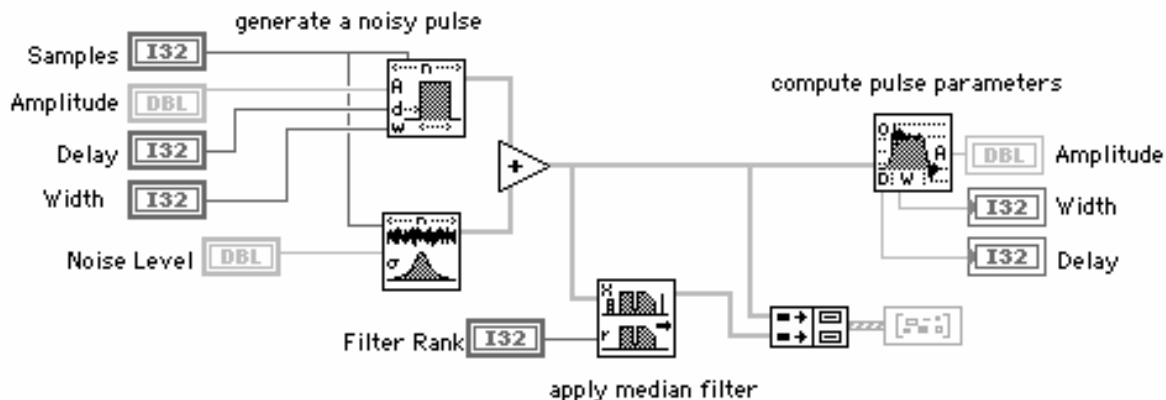
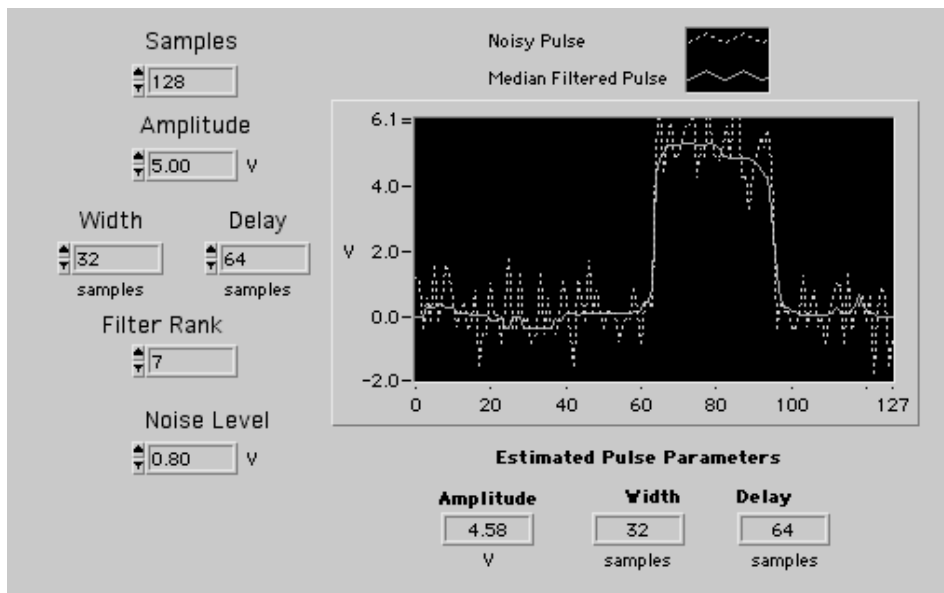
**HTML:** Hypertext Markup Language, a language consisting of codes embedded in text documents stored on "server" computers connected to the World Wide Web, and providing instructions to any "client" computers about how the document should be displayed, and how to link up to other documents on the web. A sample of code from a Web page:

```
<HTML>""<HEAD>"<TITLE>Molecular Structure</TITLE>"<IMG ALIGN=middle
SRC="gifs/at_work.gif"> This page under
construction!"</HEAD>"<BODY>"<CENTER><H2>Determination of the bR's
Structure</H2></CENTER>" We have already seen that bR is a bundle of seven
transmembrane helices."This structural feature can also be found in other
integral membrane receptor proteins, such as those of the adrenergic,
cholinergic, serotonin, somatostatin, and other receptor families.
<A HREF="badrefs.htm#R">Rao et. al.</A> have proposed that "...the packing
of seven helices together may represent a uniquely stable arrangement that
has been achieved through a process of convergent evolution." The
current model of bR's structure, is based on Henderson's et. al. work using
electron cryo-microscopy, and a refinement proposed by Chou et. al. using an
energy-based approach.<P> "<HR size=4> <CENTER> <H4>Model for the Structure of
Bacteriorhodopsin Based on High-Resolution Electron Cryo-Microscopy <A
<LI><B>Energy:</B> all-trans &lt 13-cis by about 10 kcal/mol""<UL>"<LI><A
HREF="badhome.htm">Home</A>"<LI><A
HREF="badintro.htm">Introduction</A>"</UL>"</BODY>"</HTML>""
```

**JAVA:** A high level platform-independent, object-oriented programming language similar in structure to C++, but designed especially to work with HTML documents on the World Wide Web. Java applets are programs which execute automatically on the machine of any user who visits the page, thus allowing the page to display animations and provide interactivity. A sample piece of code:

```
if (g == null) {
    Xmin = 1E20f;
    Ymin = 1E20f;
    Xmax = -1E20f;
    Ymax = -1E20f;
    turtle = new CLSTurtle(startAngle, 0, 0, 0, 0, 1, 1);
} else {
    float frwidth = Xmax - Xmin;
    if (frwidth == 0)
        frwidth = 1;
    float frheight = Ymax - Ymin;
    if (frheight == 0)
        frheight = 1;
    float xscale = (size().width - border * 2 - 1) / frwidth;
    float yscale = (size().height - border * 2 - 1) / frheight;
    int xoff = border;
    int yoff = border;
    if (normalizescaling) {
        if (xscale < yscale) {
            yoff += ((size().height - border * 2)
                - ((Ymax - Ymin) * xscale)) / 2;
            yscale = xscale;
        } else if (yscale < xscale) {
            xoff += ((size().width - border * 2)
                - ((Xmax - Xmin) * yscale)) / 2;
            xscale = yscale;
        }
    }
}
```

**LabView** is a graphical programming language designed for performing tests and measurements, acquiring data, mathematically analyzing it and presenting the graphic results. This software is used widely in process control and factory automation, and for controlling biomedical laboratory data acquisition. The visual user interface can display a wide variety of real measurement instruments (like oscilloscopes or chart recorders, stimulators, spectrum analyzers, etc.). To describe your system's behavior, you assemble block diagrams and interconnect them logically for the correct flow of data. Thousands of real-world instruments can be specified in LabView models, and there are specialized software add-on libraries for image analysis and signal processing tasks. Here is a sample runtime display of a signal, and a sample program (i.e. wiring diagram of modules).



**LISP:** An acronym for LISt Processing. A high-level language designed to process data consisting of list. It is especially suited for text manipulation and analysis. A popular language used for solving artificial intelligence problems, like PROLOG. There are computers with architectures optimized to run LISP software (LISP workstations). The well-known 'computer psychiatrist' program ELIZA is written in LISP. A sample of LISP code:

```
(DEFINE
 (FACTORIAL X)
 (PROG (F)
 (SETQ F 1)
 AGAIN
 (COND
 ((ZEROP X)
 (RETURN F)))
 (SETQ F(TIMES F X))
 (SETQ X (SUB1 X))
 (GO AGAIN)))
```

**MatLab:** this is a high performance language for technical computing, particularly in engineering and computational sciences. It integrates computation, visualization and programming into an easy-to-use environment where problems are expressed in familiar mathematical notation. Typical uses include math and computation, algorithm development, modeling, simulation, prototyping, data analysis, visualization, and graphical user interface design.

A sample of MatLab source code:

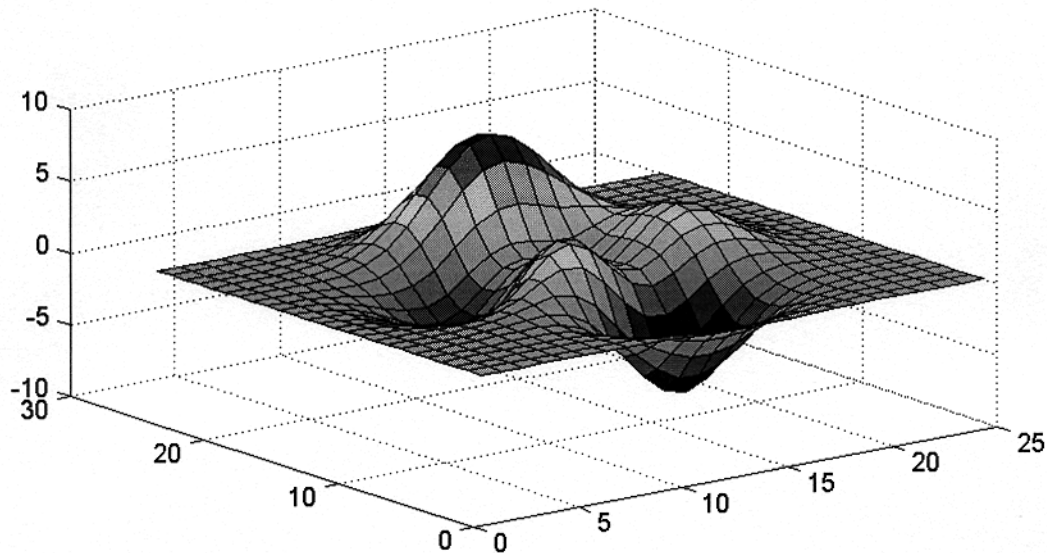
```
elseif strcmp(action,'eval'),
 % Assemble and execute the completed command
 hndlList=get(gcf,'UserData');
```

```

n=length(hndlList);
mcwHndl=hndlList(1);
evalStr=str2mat(' z=peaks(25);');
for count=2:n,
    newStrList=get(hndlList(count),'UserData');
    newStrVal=get(hndlList(count),'Value');
    newStr=deblank(newStrList(newStrVal,:));
    if ~isempty(newStr), evalStr=str2mat(evalStr,newStr); end;
end;
set(mcwHndl,'String',evalStr);
evalmcw(mcwHndl);

```

A sample graphic from a MatLab program:



**Modula-2:** A high level programming language developed by Niklaus Wirth as a replacement for Pascal; similar to Pascal. Thought to have the potential to become a leading programming language. A sample of the code:

```

module Prime; from InOut import writeln, writecard, writestring;
const
    Size=8190;
var
    Flags      :array[0..Size] of Boolean;
    I,K, Prime  :Cardinal;
    Count,     Inter :Cardinal;
begin
    writeln;
    writestring('10 iterations');
    for Inter:=1 to 10 do
        Count:=0;
        for I:=0 to Size do
            Flags[I]:=True
        end;
        for I:=0 to Size do
            if Flags[I] then
                Prime:=I+I+3;
                K:=I + Prime
                while K = Size do
                    Flags[K]:=False;
                    Inc(K,Prime);
                end;(*while*)
                Inc(Count);
            end; (*if*)
        end; (*for I)
    end; (*for Inter*)
    writeln;
    writecard(Count,6);

```

```

        writestring(' primes'),
    end Prime.

```

**PASCAL:** A high-level language named in honor of Blaise Pascal, french mathematician (1623-1662), designed to support the concept of structured , orderly top-down programming. Widely taught in beginning programming courses (dialects like Macintosh Pascal by Symantec and Turbo Pascal by Borland). A sample:

```

program BullsEye;
const
    nCircles = 100;
    vTotal = 200;
    hTotal = 200;
var
    i : integer;
    hIncr : integer;
    vIncr : integer;
begin
    hIncr := hTotal div (2 * nCircles);
    vIncr := vTotal div (2 * nCircles);
    PaintOval(0, 0, vTotal, hTotal);
    for i := 1 to nCircles do
        InvertOval(vIncr * i, hIncr * i, vTotal - vIncr * i, hTotal - hIncr *
i)
    end.

```

**Perl:** created by Larry Wall in the mid-80's, the Practical Extraction and Report /Language is a text file processing language considered somewhat easier to use than C; it is particularly strong in its ability to fetch input from Web users and in displaying parts of web-based databases. It is a "free" open-source code language supported on Unix and Linux-based computers as well as on the PC and the Macintosh. It is currently the language of choice for bioinformaticists. A sample piece of code:

```

#!/usr/local/bin/perl
#nucleotide to amino acid translation.
$minlength = 3;
$reading frames = 3;
$unknown = ``UNK``;
#retrieve command line parameter.
$input = @ARGV(0);
if length($input)<$minlength)
die ( ``$0: Place the nucleotide string on the command line.\n\n" );
) # if
printf( ``Nucleotide sequence: $input\n\n" ); etc.

```

**PL/1:** A programming language developed by IBM; abbreviation for Programming Language One. A structured language; a version or dialect used in teaching programming is PL/C. A sample of code:

```

1 PAY:PROC OPTIONS (MAIN);
2 DCL PAYNO FIXED(5),
    HOURS FIXED(2),
    RATE FIXED(4,2),
    TAX FIXED(5,2),
    GROSS FIXED(5,2);
3 GET LIST(PAYNO,HOURS,RATE,TAX);
4 GROSS=(HOURS*RATE)-TAX;
5 PUT LIST(PAYNO,HOURS,RATE,GROSS);
6 END;

```

**PILOT:** Acronym for Programmed Inquiry, Learning or Teaching. This is a specialized application language to aid development of Computer-Assisted Instruction (CAI) software, and, in particular, CAI tutorials. The language is composed of powerful, nearly syntax-free conversation-processing statements. A sample of code:

```

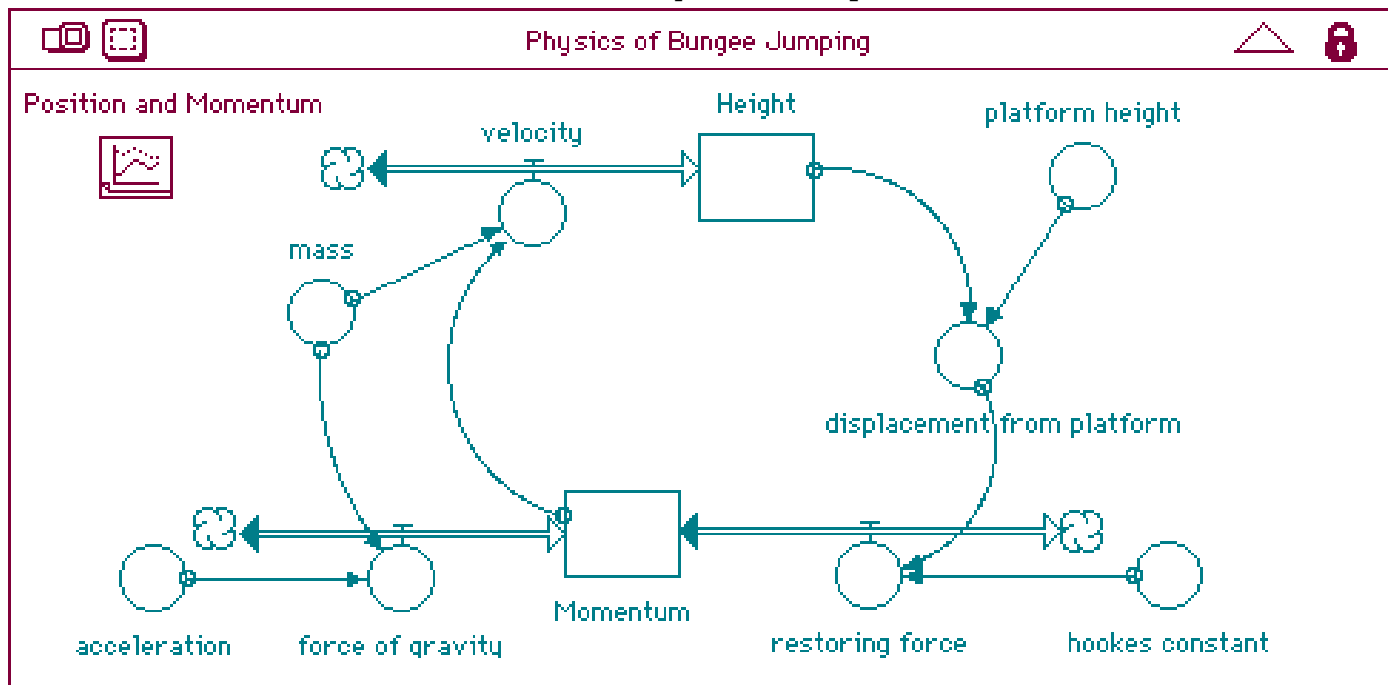
PR: U
T: What protein is found in thick myofilaments?
A:
M: myosin
TY: Well done! Myosin it is!
TN: Sorry, the correct answer is myosin.

```

**PROLOG:** Acronym for PROgramming in LOGic; a high-level logic-based language created by Alain Colmerauer, University of Marseilles. Used in the field of artificial intelligence. PROLOG is designed to manipulate knowledge instead of numbers. It is composed mainly of common English words; these words are used to state facts, relationships and patterns in a logical, concise way. What distinguishes the language's structure from traditional languages is that each line of code contains both instructions to, and data to be manipulated by, the computer. A sample of code:

```
X is-route-between (Y Z) if
  X begins-with Y &
  X ends-with Z &
  X is-connected
(X1Y) begins-with X
X ends-with Y if
  X reverses-to Z &
  Z begins-with Y
(X) is-connected
(X1 X21Y) is-connected if
  X1 linked-to X2 &
  (X21Y) is-connected
X linked-to-Y if X joined-to Y
X linked-to Y if Y joined-to X
```

**STELLA:** Stella is a "programming environment" designed to allow construction of, and interaction with, computer simulations. The environment consists of a high-level mapping and input/output layer, and a model construction layer, with an assortment of graphical icon-based modeling tools which can be interconnected to model dynamic processes. Many of the mathematical relationships between interconnected objects (Stella is another object-oriented programming language) are generated automatically. Shown below is the screen model showing linked elements like stocks and flows, followed by the script of the same model.



```
Physics of Bungee Jumping
Height(t) = Height(t - dt) + (velocity) * dt
INIT Height = platform_height
INFLOWS:
velocity = Momentum/mass
Momentum(t) = Momentum(t - dt) + (force_of_gravity - restoring_force)
* dt
INIT Momentum = 0
INFLOWS:
force_of_gravity = mass*acceleration
OUTFLOWS:
restoring_force = hookes_constant*displacement_from_platform
acceleration = -9.8
```

```
displacement_from_platform = Height-platform_height  
hookes_constant = 20  
mass = 75  
platform_height = 100
```