**Language comparison of common constructs**. I have abstracted much of this to show commonality. I do not guarantee all of this is perfectly accurate. e.g. my FORTRAN knowledge is f77 and I know there are new things that call themselves FORTRAN that include all of these constructs. The line for “objects” is intentionally simple. The entire comparison of object oriented programming ideas would require another even larger table to compare features and some subtle details.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | FORTRAN | C/C++ | matlab | Java | csh | perl | python |
| if | IF ()  statements  ELSEIF()  statements  ELSE  statements  ENDIF | if() {  statements  }else if (){  statements  }else{  statements  } | if( exp)  commands  else if()  commands  else  commands  end | if() {  statements  }else if (){  statements  }else{  statements  } | if()  commands  else if  commands  else  commands  end | if() {  statements  }elsif (){  statements  }else{  statements  } | if exp  statements  elif  statements  else:  statements |
| switch-case | NA | switch(){  case:  statements;  case:  statements;  default:  statements;  } | switch exp  case {exp}  commands  case {exp}  commands  otherwise  commands  end | switch(){  case:  statements;  case:  statements;  default:  statements;  } | switch  case xx  commands  case yy  commands  default:  commands  endsw | switch(){  case {  statements;  } case {  statements;  } | na |
| for / do loop | DO sn exp  statements  sn continue | for(x;y;z){  statements  } | f or loopdef  commands  end | for(x;y;z){  statements  } | foreach x ()  commands  end | for(x;y;z){  statements  }  or  foreach () {  statements  }  or  loop | for loopdef  statments |
| while loop | while(exp)  statements  endw | while(exp){  statements  } | while (exp)  statements  end | while(exp){  statements  } | while(exp)  statements  end | while(exp){  statements  }  or  unitl(){  statements  } | while exp  statements |
| do-while loop | na | do {  statements  } while (); | na | do {  statements  } while (); | na | na | na |
| break | GOTO | break | break | break | break or goto | break | break |
| continue | GOTO | continue | continue | continue | continue or goto | next | continue |
| transfer control to procedure | call  or  x=func(args) | x=func(args)  or  func(args) for void return | x=func(args) | x=func(args)  or  func(args) for void return | func args  return with global variables | func(args)  return with global variables | x=func(args)  or  func(args) |
| procedure definition | subroutine  statements  end  or  T function(n)  statements  end | T name(args)  {  statements  } | function T=name(args)  statements  terminate by eof – each procedure must be 1 file | T name(args)  {  statements  }  java style prefers objects | statements  Notes:   1. one procedure per file 2. Some other shells allow procs | sub name (args) {  statements  } | def proc(args)  statements |
| data structures | na in f77  ratfor or f90 have this | struct {  declarations  } | s=struct(args) | struct {  declarations  } | na | several predefined types | several predefined types |
| objects | no | C++ only | yes | core idea | no | yes | yes |