Fortran 202X progress & interim "Benefits of standardisation" survey results

A. Shterenlikht

Mech Eng Dept, The University of Bristol, Bristol BS8 1TR, UK, mexas@bristol.ac.uk

2018 Fortran Specialist Group (FSG) annual meeting, BSC offices, London, 27-SEP-2018

Fortran 202X feature survey results: ▸ N2147

Closed on 23-JAN-2018. 137 responses, pre-set choices:

- 6.21 Generic Programming or Templates
- 5.04 Automatic Allocation on READ into ALLOCATABLE character or array
- 4.91 Block Oriented or Structured Exceptions
- 4.48 Unsigned INTEGER
- 3.64 Easier Manipulation of NUL-terminated strings
- 3.58 Bit Strings
- 3.52 Conditional Expressions (like C's ? operator)

Lots of other requests

Key point: your opinion has been heard!

J3 FEB-2018 meeting - analysis of survey results

- 2 *Large* features were accepted by J3 straight away: (1) *Generic programming or templates* and (2) *Exceptions* ▸ 18-122r1
- Each participant was then asked to propose up to another 5 features, based on N2141 and their personal views.
- After extensive discussion J3 made their recommendations to WG5 ▸ 18-156

- Nobody argued for *Conditional Expressions* so this is out.
- *Bit strings* are approved as a *Medium/Large* feature. Could replace *Unsigned integers*
- *NUL-terminated strings* are approved as a *Small* feature
- *Automatic Allocation on READ* failed to get a majority.
- 4 undecided *Small* features deferred to WG5

Key point: your opinion has been taken into account!

Why did Automatic Allocation on READ fail

- One J3 member argued for this ▸ 18-135
- Subgroup recommendation and the straw vote (SV): ▸ minutes215

```
18-135

"Use cases for unallocated list item in READ statement"
This was broken out from allocatables in ERRMSG, IOMSG.
Discussion items:  Definite performance issue, as it
would require double-reading.   Appreciate the goal, but
seems unrealistic.  List directed could be a problem.
Format reversion would be a problem. Simple would be
valuable, but complicated could be very complicated

SV:  0 - 8 - 4
```

JUN-2018 WG5 meeting

- *Generic programming or templates* and *Exceptions* were accepted.
- Of the 4 undecided J3 features WG5 wants to pursue only 1: ▸ N2158

```
- += and *= syntax
  straw vote: yes:  3 - no: 11 - undecided:  5
- constructors for derived types
  straw vote: yes:  1 - no:  8 - undecided: 12
- allow a specification section in some constructs
  without block/endblock
  straw vote: yes:  4 - no:  8 - undecided:  7
- allow arrays in allocatables with coarray components
  straw vote: yes:  8 - no:  5 - undecided:  7
```

Summary of WG5 2018 resolutions: ‣ N2152

```
    RESOLUTIONS OF THE WG5 MEETING
ON JUNE 11 TO 15, 2018 IN BERKELEY, CA, USA

B6.  Content of and schedule for revision of Fortran 2018

WG5 encourages members in all countries to read WG5-N2147,
which contains the results of a worldwide survey on
possible future language developments, to read PL22.3
paper 18-156.txt which reviews the content of WG5-N2147
and to contribute further requirements during the coming
months.  The intention is that it should be possible
to define the features to be in the next revision at the
WG5 meeting in August 2019.
```

Strategic plan - ▸ N2154

Started planning further revision	2017-06	done
Choose issues that need attention	2018-06	done
Preliminary choice of technical content	2019-08	Tokyo, Japan
Final choice of technical content	2020-06	St Paul, US
CD constructed	2021-06	Manchester
CD ballot initiated	2021-07	

Stan Whitlock 1948-2018

▸ Obituary by Steve Lionel

- Joined DEC in 1976 (then Compaq, Intel) - Fortran compiler lead
- J3 at least from 1992 to 2016: GEN (General Concepts), head of JOR, INTERP, defect reports, secretary

"Benefits of standardisation" - interim survey results

Benefits of continuing Fortran standardisation, FSG survey, widely advertised, start 30-JUL-2018, end 31-DEC-2018. ▸ Full interim report
Please participate: ▸ https://goo.gl/forms/JUFUReOoVUin2m8D2

Q1. Have newer Fortran standards brought you any of the following benefits?

0 - No cost saving; 5 - Huge cost saving

```
!Potential benefit                     #resp   4,5
Better code expressiveness              371    84%
Improved code modularity                371    78%
Cut debugging costs                     369    63%
Better optimised code                   367    62%
Better interoperability                 371    59%
Cut development costs                   365    52%
Ability to target new architectures     361    48%
Cut deployment costs                    364    46%
Cuts maintenance costs                  365    46%
!Other                                  49
```

Q2. Features you use - F95 - 365 responses

do/end do loop	345	95%
whole array operations	327	90%
dynamic memory allocation	329	90%
free form syntax	324	89%
modules	326	89%
implicit none	326	89%
array sections	295	81%
exit, cycle	282	77%
module procedures	276	76%
intrinsic procedures for arrays	255	70%
optional arguments	255	70%
select case	248	68%
allocatable components	236	65%
pointers	219	60%
internal and recursive procedures	217	60%
generic interfaces	212	58%
cpu_time	201	55%
operator overloading	157	43%
null	153	42%
parametric intrinsic types	134	37%
multibyte characters	58	16%
!Other: where, forall		

F2003 - 314 responses

C interop	277	72%
OS: envars, command line, etc.	178	57%
inheritance	163	52%
dynamic type allocation	161	51%
type extension	156	50%
type—bound procedures	152	48%
polymorphism	142	45%
procedure pointers	136	43%
flush	128	41%
input_unit, output_unit, error_unit	125	40%
FPE	120	38%
PDT	99	32%
stream IO	99	32%
explicit type in array constructor	86	27%
deferred type parameters	83	26%
finalisers	73	23%
async IO	55	18%
DTIO	52	17%
control of rounding modes	45	14%
volatile	24	8%

```
!Other: protected attribute, bit manip.
!associate, abstract types, 63 char length limit
```

F2008 - 300 responses

int8 , int16 , int32 , int64 , real32 ...	167	56%
64−bit integer	145	48%
Bessel and err. func, e.g. BESSEL_J0	116	39%
execute_command_line	114	38%
submodules	112	37%
do concurrent	109	36%
c_size_of	98	33%
contiguous	95	32%
newunit	95	32%
coarrays + coar. intrinsics	95	32%
findloc − array searching	95	32%
bit manipulation func: bitwise comp	89	30%
compiler_version , compiler_options	84	28%
block construct	84	28%
new complex intrinsics: ACOS, ACOSH...	76	25%
HYPOT, NORM2	71	24%
storage_size	70	23%
%re , %im shorthands	69	23%
more complex intrinsics	65	22%
initial pointer association	62	21%
impure elemental procedures	50	17%
atomics	48	16%
critical	45	15%
locks	41	14%
max array rank of 15	38	13%

```
!Other: mold, error stop, ...
```

F2018 (use or want to use) - 160 responses

```
assumed rank: select rank                        76      48%
assumed type                                     67      42%
improved IEEE support                            67      42%
ISO_Fortran_binding.h, ...                       65      41%
C desc for assumed shape dummy                   56      35%
collectives                                      41      26%
new atomics                                      33      21%
teams                                            31      19%
events                                           30      19%
image failure                                    21      13%
!Other: error stop in pure procedures,
!too recent, exceptions
```

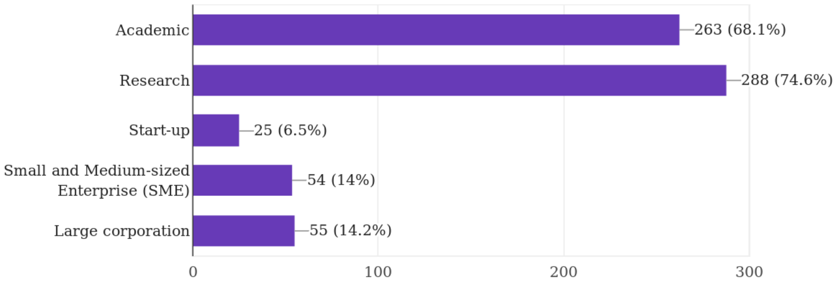
Q3: Future of the Fortran language

- If you think Fortran is lacking particular features which would help you, please detail them here.
  134 responses.
- What Fortran compiler(s) are you using? And does it (do they) support the standard features you want to use?
  321 responses.
  Compilers: gfortran, ifort, Cray, PGI, NAG, Flang, g95, OpenUG, Lahey, Sun, Open Watcom F77.
- Would you like to know more about the work of the Fortran standards committee?
  NO 59%, YES 41%.

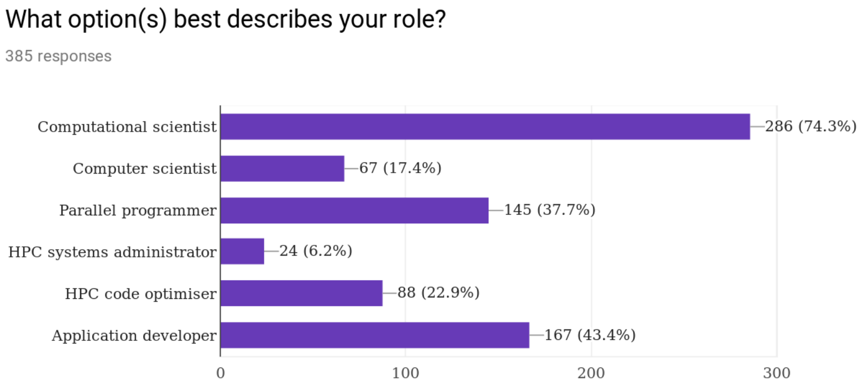
Q4: About you: Industrial sectors

Select the industries that apply to you:

386 responses



What option(s) best describe your role?



Which country are you based in?

111	32%	United States
107	31%	United Kingdom
25	7%	Germany
14	4%	Netherlands
8	2%	France
6	2%	Canada, Italy
5	2%	Russia, Sweden
4	1%	Australia, Brazil, New Zealand, Spain
3	1%	Argentina, Belgium, China, Norway
2	1%	Gambia, Greece, India, Poland, Switzerland, Vietnam
1	<0.5%	Algeria, Ashmore & Cartier Islands, Cayman Islands, Chile, Czech Republic, Denmark, Estonia, Finland, Iran, Ireland, Luxembourg, Malaysia, Romania, Saudi Arabia, Slovenia, Ukraine.
343		TOTAL