WIKIPEDIA

Variable-length array

In computer programming, a **variable-length array** (**VLA**), also called **variable-sized** or **runtime-sized**, is an array data structure whose length is determined at run time (instead of at compile time).[1] In C, the VLA is said to have a variably modified type that depends on a value (see Dependent type).

The main purpose of VLAs is to simplify programming of numerical algorithms.

Programming languages that support VLAs include Ada, Algol 68 (for non-flexible rows), APL, C99 (although subsequently relegated in C11 to a conditional feature, which implementations are not required to support;[2][3] on some platforms, could be implemented previously with `alloca()` or similar functions) and C# (as unsafe-mode stack-allocated arrays), COBOL, Fortran 90, J, and Object Pascal (the language used in Borland Delphi and Lazarus, that uses FPC).

Contents

Memory

Allocation

- The GNU C Compiler allocates memory for VLAs with automatic storage duration on the stack.[4] This is the faster and more straightforward option compared to heap-allocation, and is used by most compilers.
- VLAs can also be allocated on the heap and internally accessed using a pointer to this block.

Implementation

C99

The following C99 function allocates a variable-length array of a specified size, fills it with floating-point values, and then passes it to another function for processing. Because the array is declared as an automatic variable, its lifetime ends when `read_and_process()` returns.

```
float read_and_process(int n)
{
    float vals[n];

    for (int i = 0; i < n; ++i)
        vals[i] = read_val();

    return process(n, vals);
}
```

In C99, the length parameter must come before the variable-length array parameter in function calls.[1] In C11, a `__STDC_NO_VLA__` macro is defined if VLA is not supported.[5] GCC had VLA as an extension before C99.

Linus Torvalds has expressed his displeasure in the past over VLA usage for arrays with predetermined small sizes because it generates lower quality assembly code. [6] With the Linux 4.20 kernel, Linux kernel is effectively VLA-free.[7]

Although C11 does not explicitly name a size-limit for VLAs, some readings believe it should have the same maximum size as all other objects, i.e. SIZE_MAX bytes.[8] However, this reading should be understood in the wider context of environment and platform limits, such as the typical stack-guard page size of 4 KiB, which is many orders of magnitude smaller than SIZE_MAX.

Ada

Following is the same example in Ada. Ada arrays carry their bounds with them, so there is no need to pass the length to the Process function.

```
type Vals_Type is array (Positive range <>) of Float;

function Read_And_Process (N : Integer) return Float is
   Vals : Vals_Type (1 .. N);
begin
   for I in 1 .. N loop
      Vals (I) := Read_Val;
   end loop;
   return Process (Vals);
end Read_And_Process;
```

Fortran 90

The equivalent Fortran 90 function is

```
function read_and_process(n) result(o)
    integer,intent(in)::n
    real::o

    real,dimension(n)::vals
    integer::i

    do i = 1,n
       vals(i) = read_val()
    end do
    o = process(vals)
end function read_and_process
```

when utilizing the Fortran 90 feature of checking procedure interfaces at compile time; on the other hand, if the functions use pre-Fortran 90 call interface, the (external) functions must first be declared, and the array length must be explicitly passed as an argument (as in C):

```
function read_and_process(n) result(o)
    integer,intent(in)::n
    real::o

    real,dimension(n)::vals
    real::read_val, process
    integer::i

    do i = 1,n
       vals(i) = read_val()
    end do
    o = process(vals,n)
end function read_and_process
```

Cobol

The following COBOL fragment declares a variable-length array of records `DEPT-PERSON` having a length (number of members) specified by the value of `PEOPLE-CNT`:

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  DEPT-PEOPLE.
    05  PEOPLE-CNT          PIC S9(4) BINARY.
    05  DEPT-PERSON         OCCURS 0 TO 20 TIMES DEPENDING ON PEOPLE-CNT.
        10  PERSON-NAME     PIC X(20).
        10  PERSON-WAGE     PIC S9(7)V99 PACKED-DECIMAL.
```

The COBOL VLA, unlike that of other languages mentioned here, is safe because COBOL requires one to specify the maximal array size – in this example, `DEPT-PERSON` cannot have more than 20 items, regardless of the value of `PEOPLE-CNT`.

C#

The following C# fragment declares a variable-length array of integers. Prior to C# version 7.2, a pointer to the array is required, requiring an "unsafe" context. The "unsafe" keyword requires an assembly containing this code to be marked as unsafe.

```
unsafe void DeclareStackBasedArrayUnsafe(int size)
{
    int *pArray = stackalloc int[size];
    pArray[0] = 123;
}
```

C# version 7.2 and later allow the array to be allocated without the "unsafe" keyword, through the use of the Span feature.[9]

```
void DeclareStackBasedArraySafe(int size)
{
    Span<int> stackArray = stackalloc int[size];
    stackArray[0] = 123;
}
```

Object Pascal

In this language, it is called a dynamic array. The declaration of such a variable is similar to the declaration of a static array, but without specifying its size. The size of the array is given at the time of its use.

```
program CreateDynamicArrayOfNumbers(Size: Integer);
var
  NumberArray: array of LongWord;
begin
  SetLength(NumberArray, Size);
  NumberArray[0] := 2020;
end.
```

Removing the contents of a dynamic array is done by assigning it a size of zero.

```
...
SetLength(NumberArray, 0);
...
```

References

1. "Variable Length Arrays" (https://web.archive.org/web/20180126153326/http://docs.cray.com/books/004-2179-001/html-004-2179-001/z893434830malz.html). Archived from the original (http://docs.cray.com/books/004-2179-001/html-004-2179-001/z893434830malz.html) on 2018-01-26.
2. "Variable Length – Using the GNU Compiler Collection (GCC)" (https://gcc.gnu.org/onlinedocs/gcc/Variable-Length.html).
3. ISO 9899:2011 Programming Languages – C 6.7.6.2 4.
4. "Code Gen Options - The GNU Fortran Compiler" (https://gcc.gnu.org/onlinedocs/gfortran/Code-Gen-Options.html).
5. § 6.10.8.3 of the C11 standard (n1570.pdf)
6. "LKML: Linus Torvalds: Re: VLA removal (was Re: [RFC 2/2] lustre: use VLA_SAFE)" (https://lkml.org/lkml/2018/3/7/621). *lkml.org*.
7. "The Linux Kernel Is Now VLA-Free: A Win For Security, Less Overhead & Better For Clang - Phoronix" (https://www.phoronix.com/scan.php?page=news_item&px=Linux-Kills-The-VLA). *www.phoronix.com*.
8. §6.5.3.4 and §7.20.3 of the C11 standard (n1570.pdf)
9. "stackalloc operator (C# reference)" (https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/stackalloc). Microsoft.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Variable-length_array&oldid=988820281"