Flexible Array Members in a structure in C

Flexible Array Member(FAM) is a feature introduced in the C99 standard of the C programming language.

- For the structures in C programming language from C99 standard onwards, we can declare an array **without a dimension** and whose size is flexible in nature.
- Such an array inside the structure should preferably be declared as the **last member** of structure and its size is variable(can be changed be at runtime).
- The structure must contain at least one more named member in addition to the flexible array member.

What must be the size of the structure below?

```
struct student
{
   int stud_id;
   int name_len;
   int struct_size;
   char stud_name[];
};
```

```
The size of structure is = 4 + 4 + 4 + 0 = 12
```

In the above code snippet, the size i.e length of array "stud_name" isn't fixed and is an FAM.

The memory allocation using flexible array members(as per C99 standards) for the above example can be done as:

```
struct student *s = malloc( sizeof(*s) + sizeof(char [strlen(stud_name)])  );
```

Note: While using flexible array members in structures some convention regarding actual size of the member is defined.
In the above example the convention is that the member "stud_name" has character size.

For Example, Consider the following structure:

```
Input : id = 15, name = "Kartik"
Output : Student_id : 15
         Stud_Name  : Kartik
         Name_Length: 6
         Allocated_Struct_size: 18
```

Memory allocation of above structure:

```
struct student *s =
        malloc( sizeof(*s) + sizeof(char [strlen("Kartik")]));
```

Its structure representation is equal to:

```
struct student
{
    int stud_id;
    int name_len;
    int struct_size;
    char stud_name[6]; //character array of length 6
};
```

```
// C program for variable length members in
// structures in GCC
#include<string.h>
#include<stdio.h>
#include<stdlib.h>

// A structure of type student
struct student
{
    int stud_id;
    int name_len;

    // This is used to store size of flexible
    // character array stud_name[]
    int struct_size;

    // Flexible Array Member(FAM)
    // variable length array must be last
    // member of structure
    char stud_name[];
};
// Memory allocation and initialisation of structure
struct student *createStudent(struct student *s,
                              int id, char a[])
{
    // Allocating memory according to user provided
    // array of characters
    s =
        malloc( sizeof(*s) + sizeof(char) * strlen(a));

    s->stud_id = id;
    s->name_len = strlen(a);
    strcpy(s->stud_name, a);
```

```
    // Assigning size according to size of stud_name
    // which is a copy of user provided array a[].
    s->struct_size =
        (sizeof(*s) + sizeof(char) * strlen(s->stud_name));

    return s;
}

// Print student details
void printStudent(struct student *s)
{
    printf("Student_id : %d\n"
           "Stud_Name : %s\n"
           "Name_Length: %d\n"
           "Allocated_Struct_size: %d\n\n",
           s->stud_id, s->stud_name, s->name_len,
           s->struct_size);

    // Value of Allocated_Struct_size is in bytes here
}
// Driver Code
int main()
{
    struct student *s1 = createStudent(s1, 523, "Cherry");
    struct student *s2 = createStudent(s2, 535, "Sanjayulsha");

    printStudent(s1);
    printStudent(s2);

    // Size in struct student
    printf("Size of Struct student: %lu\n",
                   sizeof(struct student));

    // Size in struct pointer
    printf("Size of Struct pointer: %lu",
                              sizeof(s1));

    return 0;
}
```

Output:

```
Student_id : 523
Stud_Name : SanjayKanna
Name_Length: 11
Allocated_Struct_size: 23
```

```
Student_id : 535
Stud_Name : Cherry
Name_Length: 6
Allocated_Struct_size: 18

Size of Struct student: 12
Size of Struct pointer: 8
```

Important Points:

1. Adjacent memory locations are used to store structure members in memory.
2. In previous standards of the C programming language, we were able to to declare a zero size array member in place of a flexible array member. The GCC compiler with C89 standard considers it as zero size array.