Exercise 9

(Question 1)

By

Sanaz Rahbari

Introduction to programming C

Lecturer

Prof.-Dr. Ulrich Hauser-Ehninger

Ilia State University

Tbilisi

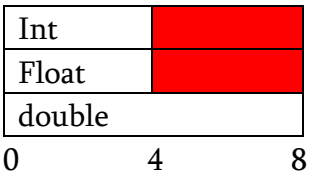
2019

```
# include <stdio .h>
int main ()
{

struct a
{
int a;
double b;
float c;
};

struct b
{
double a;
float b;
int c;
};

printf (" Size of a: %ld\n", sizeof ( struct a));
printf (" Size of b: %ld\n", sizeof ( struct b));
return 0;
}
```

Each variable has its own size, int: 4-byte, float: 4-byte and double: 8-byte. In these two structures double variable has the largest size of all and consequently needs more memory capacity as well. However the way these three variables has been set in each structure is the reason that although struct “a” and “b” have the same variables, result of size of will be different at the end.

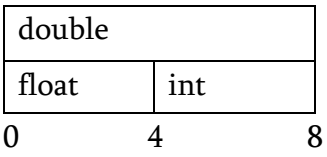
The reason is that compiler tries to align memory when we have different types of variable in structure and in order to do that it uses padding technique. In this technique compiler adds unusable memory to align structure and prevent compiler from penalty performance.

In “struct a”, the order is from int to double which is from smallest to largest size.



The table above shows “struct a” in memory which has size of 24-bytes. In the beginning as int and float each has 4-byte size there was no reason for alignment but as double added to the structure the size will be expanded and therefore padding will be needed which means each one of int and float must be expended to 8-bytes. Therefore the sum up three 8-bytes variable (after padding) will be: 3*8=24 bytes

In “struct b”, order of structure is opposite of “struct a”, in this case as double is place first so there is no need for expansion of other variables. As the sum of int and float size will be 8-byte which is equal to double size. So, the size of “struct b” will be: 8+ (2*4) = 16 bytes



References:

- https://software.intel.com/en-us/articles/coding-for-performance-data-alignment-and-structures

- https://aticleworld.com/data-alignment-and-structure-padding-bytes/

- https://www.geeksforgeeks.org/structure-member-alignment-padding-and-data-packing/