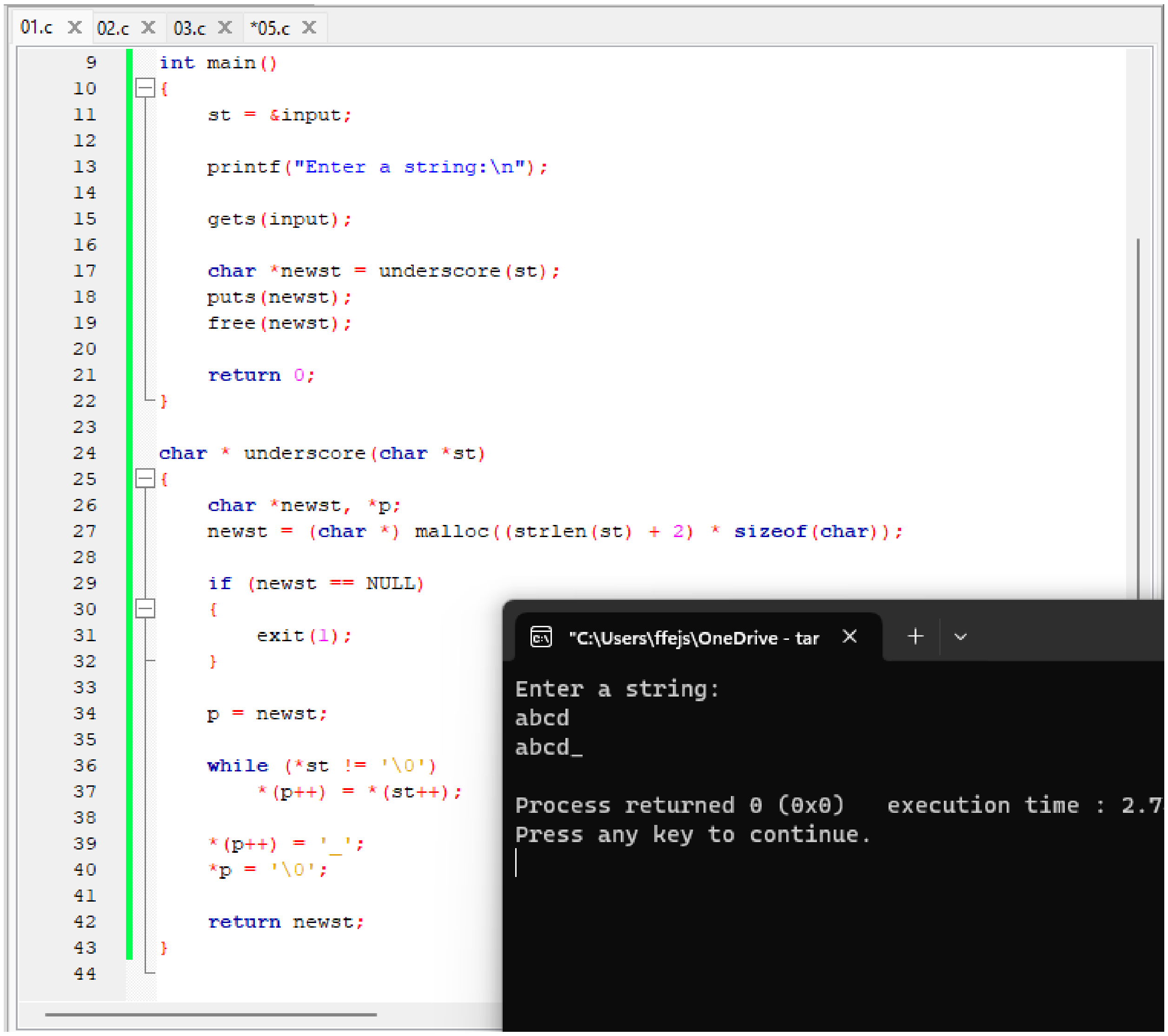
Strings as function arguments, malloc, and structures

1] Write a function **underscore()** that accepts one string **st** as argument.

- The length of **st** is unknown in advance. Use the library function **strlen()** to determine it.
    - Don't forget to include **string.h**!
- Use **malloc()** to allocate enough memory to hold a string that is one character longer than **st**.
- Copy **st** to the block of memory allocated, and add the character *underscore* _ at the end.
    - Use a loop to copy character-by-character. What type of loop?
    - Don't forget the string terminator!
- **underscore()** returns a pointer to the new string.

In the main program, ask the user to enter a string, and then print the string returned by **underscore()**; for example, if the user enters **abcd**, the program prints **abcd_**

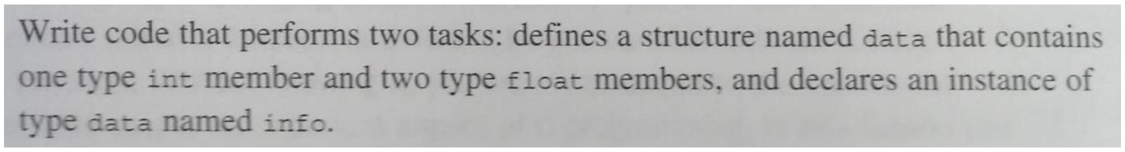
Remember to liberate the allocated string before ending the program.



```c
int main()
{
    st = &input;

    printf("Enter a string:\n");

    gets(input);

    char *newst = underscore(st);
    puts(newst);
    free(newst);

    return 0;
}

char * underscore(char *st)
{
    char *newst, *p;
    newst = (char *) malloc((strlen(st) + 2) * sizeof(char));

    if (newst == NULL)
    {
        exit(1);
    }

    p = newst;

    while (*st != '\0')
        *(p++) = *(st++);

    *(p++) = '_';
    *p = '\0';

    return newst;
}
```

```
Enter a string:
abcd
abcd_

Process returned 0 (0x0)   execution time : 2.7
Press any key to continue.
```

2] Write a program that solves Exercise 2 / p.277:

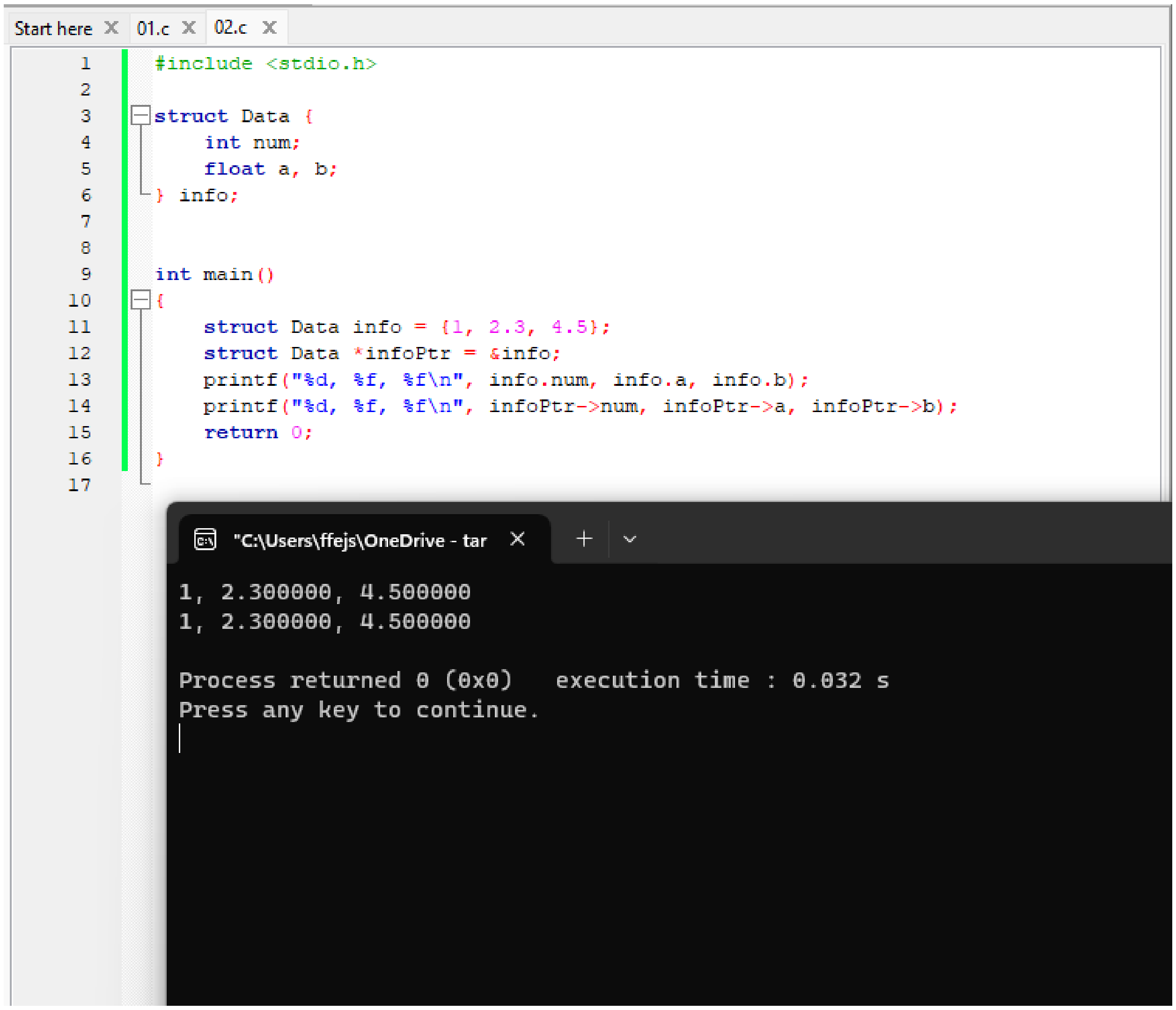


Write code that performs two tasks: defines a structure named `data` that contains one type `int` member and two type `float` members, and declares an instance of type `data` named `info`.

Initialize info with some data.

Then create a pointer **infoPtr** to **info** and print **info** two ways:

- using the variable name **info**
- using **infoPtr**

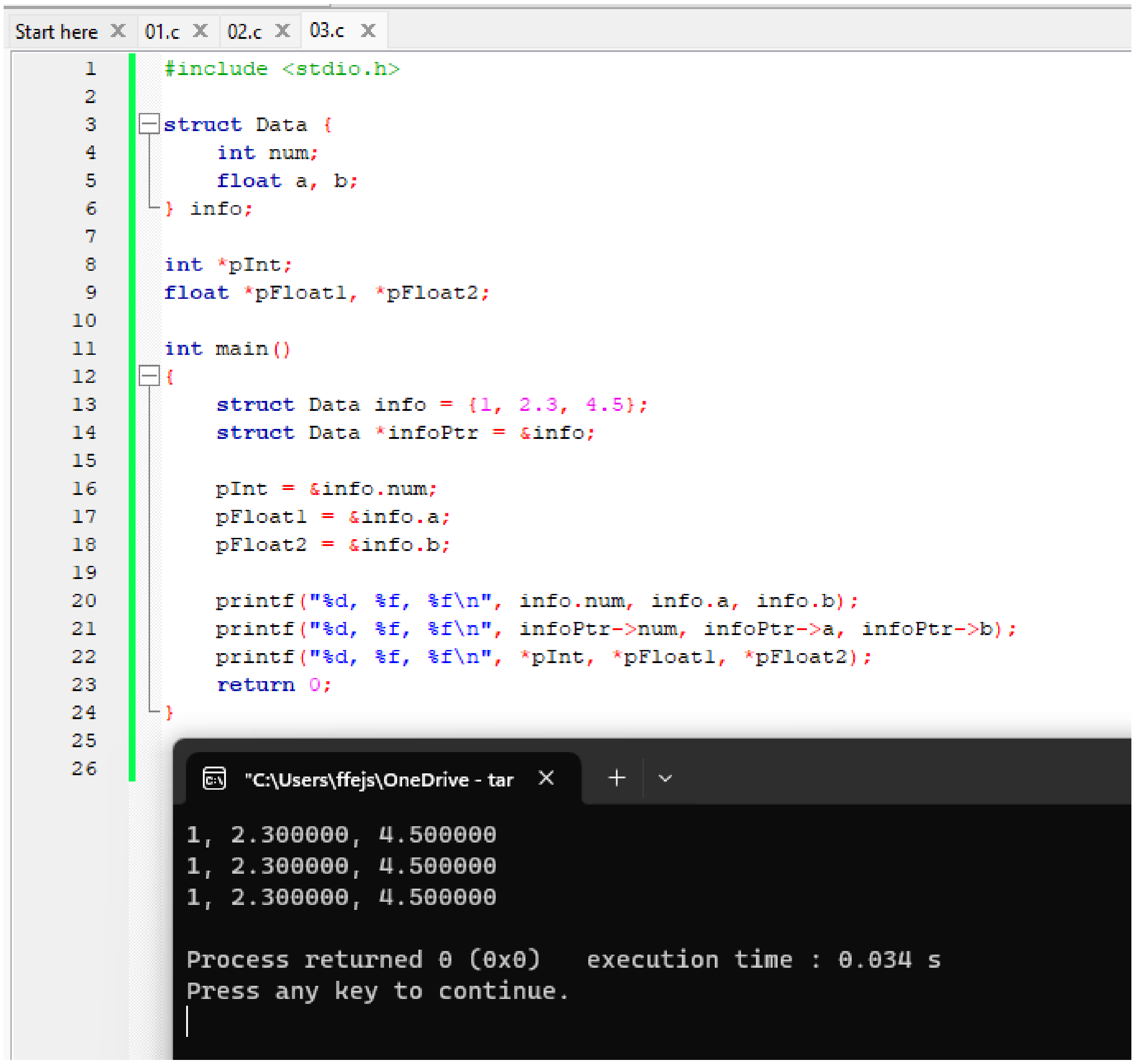


```c
#include <stdio.h>

struct Data {
    int num;
    float a, b;
} info;


int main()
{
    struct Data info = {1, 2.3, 4.5};
    struct Data *infoPtr = &info;
    printf("%d, %f, %f\n", info.num, info.a, info.b);
    printf("%d, %f, %f\n", infoPtr->num, infoPtr->a, infoPtr->b);
    return 0;
}
```

```
1, 2.300000, 4.500000
1, 2.300000, 4.500000

Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.
```

3] Using the previous program:

- Declare pInt, pFloat1, and pFloat2 to be pointers to integer and floats.
- Initialize so they will point to the integer and the float members of info.
- Print those values using the pointers.



Start here 01.c 02.c 03.c

```c
#include <stdio.h>

struct Data {
    int num;
    float a, b;
} info;

int *pInt;
float *pFloat1, *pFloat2;

int main()
{
    struct Data info = {1, 2.3, 4.5};
    struct Data *infoPtr = &info;

    pInt = &info.num;
    pFloat1 = &info.a;
    pFloat2 = &info.b;

    printf("%d, %f, %f\n", info.num, info.a, info.b);
    printf("%d, %f, %f\n", infoPtr->num, infoPtr->a, infoPtr->b);
    printf("%d, %f, %f\n", *pInt, *pFloat1, *pFloat2);
    return 0;
}
```

"C:\Users\ffejs\OneDrive - tar

```
1, 2.300000, 4.500000
1, 2.300000, 4.500000
1, 2.300000, 4.500000

Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

4] Circle all that apply:

A structure is a data type in which:

A. each element must have the same data type.

B. each element must have a pointer type only

[✓ C.] each element may have a different data type

[✓ D.] No element is defined

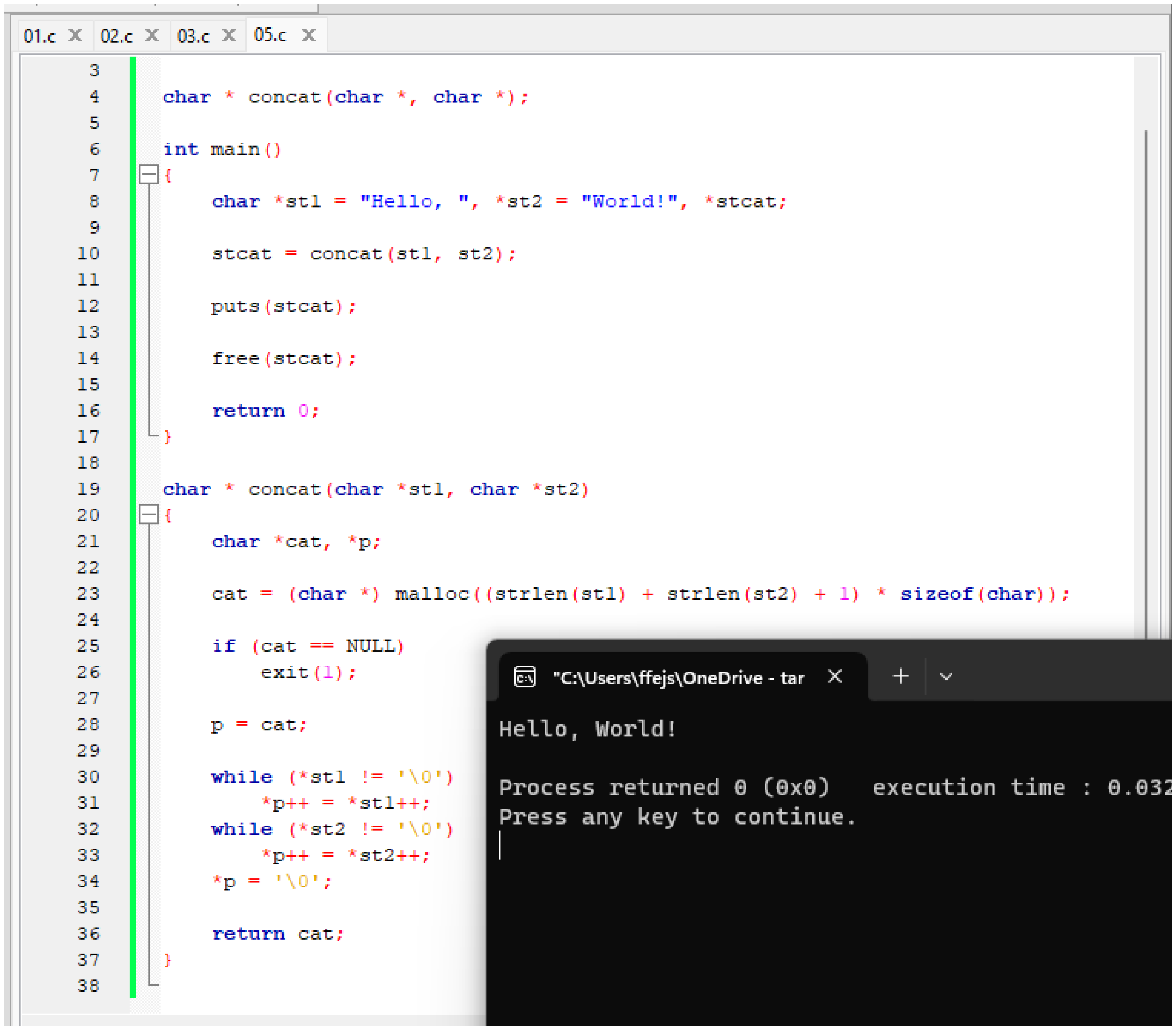
Problems for the lab report

For all problems marked with ►, take screenshots of both code and output, paste the screenshots in a report PDF file, and submit it to Canvas.

5] ► Solve Exercise 7 at the end of Ch.10:

Write a function that accepts 2 strings. Use **malloc()** to allocate enough memory to hold the 2 strings concatenated. Copy the strings to the block of memory allocated. Return a pointer to the new string. Print the new string in the main program.

Hint: Use **strlen()** to determine the lenghts of the string arguments. (Don't forget to include **string.h**!)



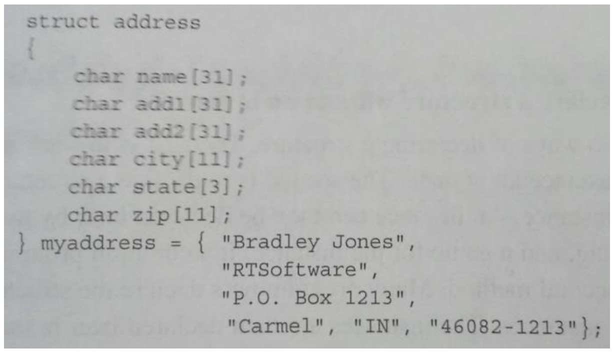
01.c 02.c 03.c 05.c

```c

char * concat(char *, char *);

int main()
{
    char *st1 = "Hello, ", *st2 = "World!", *stcat;

    stcat = concat(st1, st2);

    puts(stcat);

    free(stcat);

    return 0;
}

char * concat(char *st1, char *st2)
{
    char *cat, *p;

    cat = (char *) malloc((strlen(st1) + strlen(st2) + 1) * sizeof(char));

    if (cat == NULL)
        exit(1);

    p = cat;

    while (*st1 != '\0')
        *p++ = *st1++;
    while (*st2 != '\0')
        *p++ = *st2++;
    *p = '\0';

    return cat;
}

```

"C:\Users\ffejs\OneDrive - tar

```
Hello, World!

Process returned 0 (0x0)   execution time : 0.032
Press any key to continue.
```

6] ► Write a program that creates the structure from QUIZ 5 / p.276:

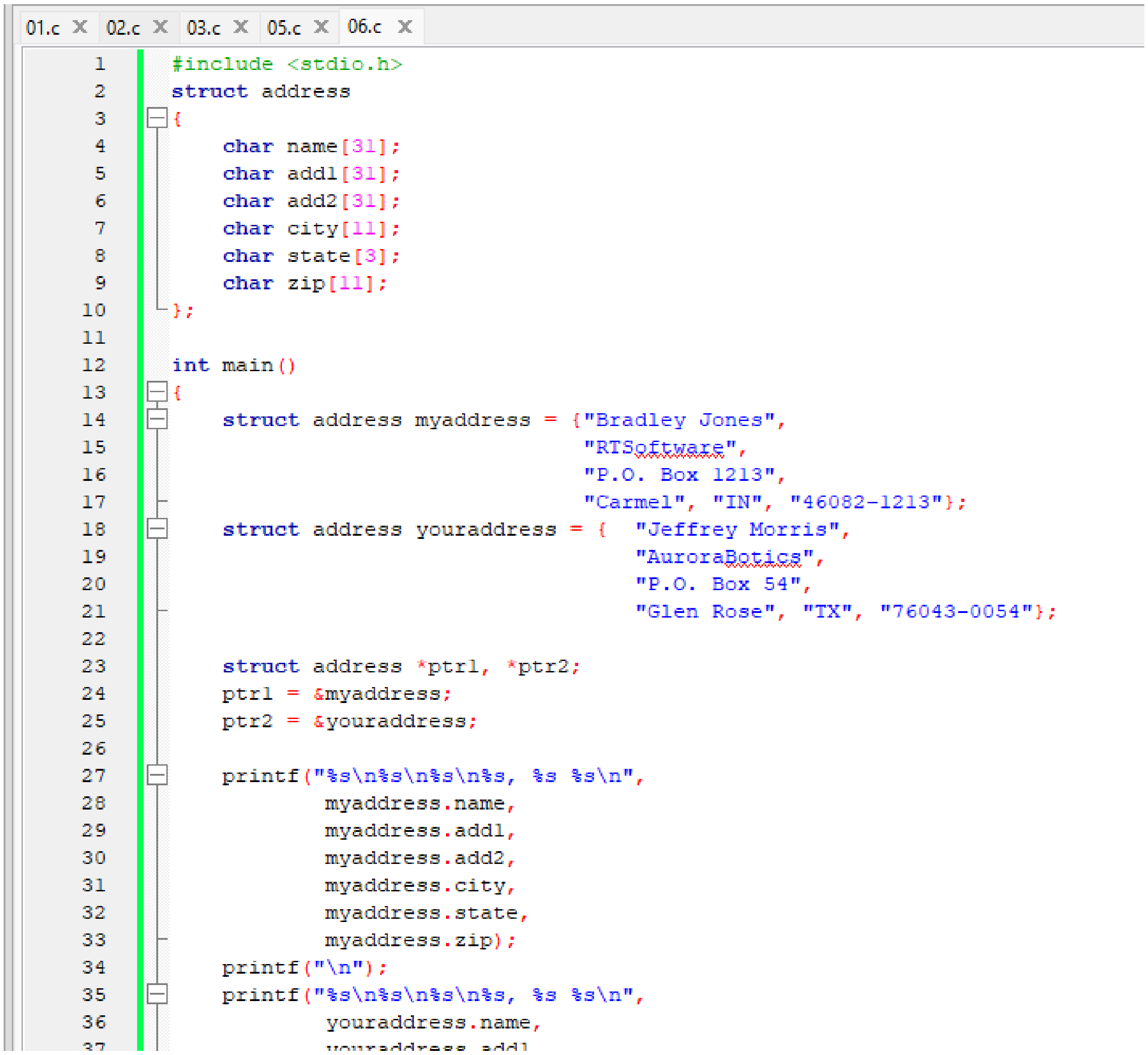


```
struct address
{
    char name[31];
    char add1[31];
    char add2[31];
    char city[11];
    char state[3];
    char zip[11];
} myaddress = { "Bradley Jones",
                "RTSoftware",
                "P.O. Box 1213",
                "Carmel", "IN", "46082-1213"};
```

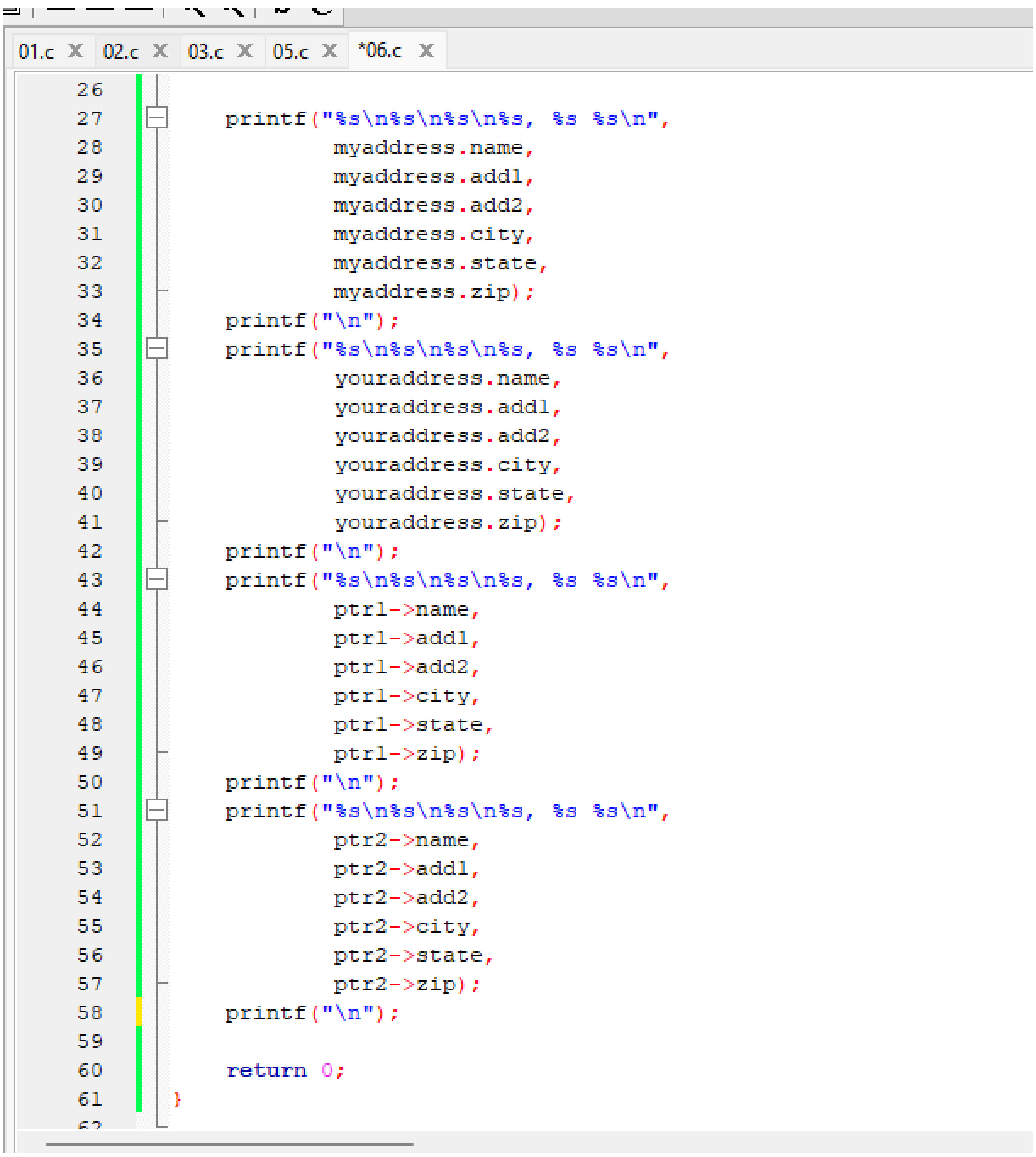
Create a second variable **youraddress** <u>of the same structure type</u>, and initialize it with data of your choice. Create two pointers **ptr1**, and **ptr2**. Make **ptr1** point to **myaddress** and **ptr2** to **youraddress**.

Print the structures two ways:

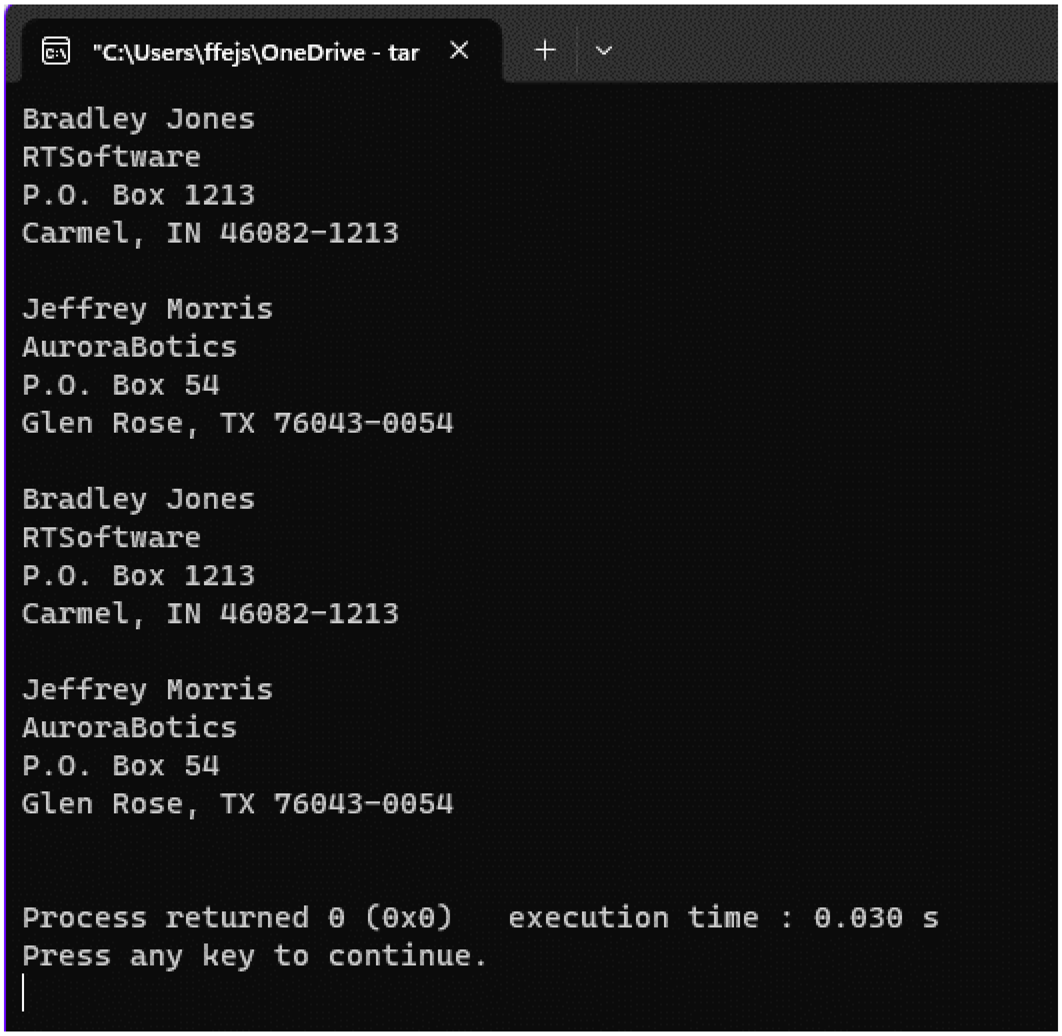
- using the variable names
- using the pointers.



```c
#include <stdio.h>
struct address
{
    char name[31];
    char add1[31];
    char add2[31];
    char city[11];
    char state[3];
    char zip[11];
};

int main()
{
    struct address myaddress = {"Bradley Jones",
                                "RTSoftware",
                                "P.O. Box 1213",
                                "Carmel", "IN", "46082-1213"};
    struct address youraddress = {  "Jeffrey Morris",
                                    "AuroraBotics",
                                    "P.O. Box 54",
                                    "Glen Rose", "TX", "76043-0054"};

    struct address *ptr1, *ptr2;
    ptr1 = &myaddress;
    ptr2 = &youraddress;

    printf("%s\n%s\n%s\n%s, %s %s\n",
            myaddress.name,
            myaddress.add1,
            myaddress.add2,
            myaddress.city,
            myaddress.state,
            myaddress.zip);
    printf("\n");
    printf("%s\n%s\n%s\n%s, %s %s\n",
            youraddress.name,
            youraddress.add1
```



```
    printf("%s\n%s\n%s\n%s, %s %s\n",
            myaddress.name,
            myaddress.add1,
            myaddress.add2,
            myaddress.city,
            myaddress.state,
            myaddress.zip);
    printf("\n");
    printf("%s\n%s\n%s\n%s, %s %s\n",
            youraddress.name,
            youraddress.add1,
            youraddress.add2,
            youraddress.city,
            youraddress.state,
            youraddress.zip);
    printf("\n");
    printf("%s\n%s\n%s\n%s, %s %s\n",
            ptr1->name,
            ptr1->add1,
            ptr1->add2,
            ptr1->city,
            ptr1->state,
            ptr1->zip);
    printf("\n");
    printf("%s\n%s\n%s\n%s, %s %s\n",
            ptr2->name,
            ptr2->add1,
            ptr2->add2,
            ptr2->city,
            ptr2->state,
            ptr2->zip);
    printf("\n");

    return 0;
}
```



```
Bradley Jones
RTSoftware
P.O. Box 1213
Carmel, IN 46082-1213

Jeffrey Morris
AuroraBotics
P.O. Box 54
Glen Rose, TX 76043-0054

Bradley Jones
RTSoftware
P.O. Box 1213
Carmel, IN 46082-1213

Jeffrey Morris
AuroraBotics
P.O. Box 54
Glen Rose, TX 76043-0054


Process returned 0 (0x0)   execution time : 0.030 s
Press any key to continue.
```

7] Circle all that apply:

If one or more members of a structure are other structures, the structure is known as:

[✓ A.] nested structure B. invalid structure

C. self-referential structure D. unstructured structure

Do all the members of a structure need to have the same size? A. Yes [✓ B. No]