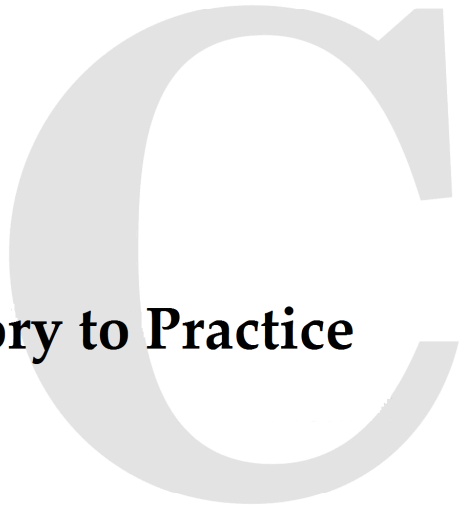
George S. Tselikis | Nikolaos D. Tselikas



C From Theory to Practice

Book's Indicative Exercises

Book's Indicative Exercises

S.1 What is the output of the following program?

```
#include <stdio.h>
int main()
{
	int i, a[] = {10, 20, 30, 40, 50};
	double b[] = {2.2, 1.94, 0.5, -1, -2};

	for(i = 0; a[i] = b[i]; i++)
		printf("%d ", a[i]);
	return 0;
}
```

S.2 Write a program that reads the integer codes of 50 products and stores them in an array, only if they have not already been stored. As such, the elements of the array must be different. The program should display them, before it ends.

S.3 What is the output of the following program?

```
#include <stdio.h>
int main()
{
	int *ptr1, *ptr2, i = 10, j = 20;

	ptr1 = &i;
	ptr2 = &j;

	ptr2 = ptr1;
	*ptr1 = *ptr1 + *ptr2;
	*ptr2 = 2*(*ptr2);
	printf("Val = %d\n", *ptr1 + *ptr2);
	return 0;
}
```

S.4 Use the pointers `p1`, `p2` and `p3` and complete the following program to read two integers and display the sum of the integers between them. For example, if the user enters `6` and `10`, the program should display `24` (`7+8+9`). The program should force the user to enter numbers less than `100` and the first integer should be less than the second.

```
#include <stdio.h>
int main()
{
      int *p1, *p2, *p3, i, j, sum;
      ...
}
```

S.5 Write a program that reads the grades of 10 students, stores them in an array and displays the maximum and the minimum grade and their positions in the array. The program should check if the input grades are within [0, 10]. Use pointer arithmetic to process the array.

S.6 What are the values of `arr` elements in the following program?

```
#include <stdio.h>
int main()
{
      int *ptr, arr[5] = {20};

      for(ptr = arr+1; ptr <= arr+4; ptr++)
            *ptr = *(ptr-1) + *(ptr+1) + 1;
      return 0;
}
```

S.7 Write a program that uses a **`for`** loop to display all lower-case letters in one line, all upper-case letters in a second line and all characters that represent the digits `0-9` in a third line.

S.8 What is the output of the following program?

```
#include <stdio.h>
int main()
{
      char str1[] = "test", str2[] = "test";
      (str1 == str2) ? printf("One\n") : printf("Two\n");
      return 0;
}
```

S.9 Write a program that forces the user to enter a string with more than 5 characters and less than 100 characters and displays it. Don't use `gets()`.

S.10 What is the output of the following program?

```
#include <stdio.h>
#include <string.h>
```

```
int main()
{
	char str[10] = "test";
	printf("%d %s\n", *strcpy(str, "n")**strcpy(str+2, "xt"),
str);
	return 0;
}
```

S.11 Write a function that takes as parameters two pointers to floats of type **double** and uses them to return a pointer to the float with the greater value. Write a program that reads two floats and uses the function to display the greater.

S.12 What is the output of the following program?

```
#include <stdio.h>
#include <string.h>
int main()
{
	char *ptr, str[] = "Text";
	int i;

	ptr = str;
	for(i = 0; i < strlen(str); i++)
	{
		printf("%c", ptr[i]);
		ptr++;
	}
	return 0;
}
```

S.13 Write a function that takes as parameters an array which contains the prices of some products in a shop and its size and returns the minimum, the maximum and the average of the prices. Write a program that reads the prices of up to 100 products and stores them in an array. If the user enters `-1`, the insertion of prices should terminate. The program should use the function to display the minimum, the maximum and the average of the prices.

S.14 What is the output of the following program?

```
#include <stdio.h>

void test(int *arg);

int var = 100;
```

```
int main()
{
	int *ptr, i = 30;

	ptr = &i;
	test(ptr);
	printf("Val = %d\n", *ptr);
	return 0;
}

void test(int *arg)
{
	arg = &var;
}
```

S.15 What is the output of the following program?

```
#include <stdio.h>

int a = 4;

int main()
{
	if(a == 0)
		return 0;
	else
	{
		printf("%d ", a--);
		main();
	}
	return 0;
}
```

S.16 Write a program that reads its command line arguments and allocates memory to store their characters in reverse order. For example, if the arguments are `next` and `time`, the program should store into the memory `txen` and `emit`.

S.17 Define a structure type named `city` with fields: name, country, and population. Write a program that uses this type to read the data of 5 cities and stores them in an array of such structures. Then, the program should read the name of a country and a number and display the country's cities whose population is greater than the input number.

S.18 Suppose that each line of the `students.txt` file contains the names of the students (assume each name is less than 100 characters) and their grades (read them as **`double`**) in two lessons like this:

```
John        Morne        7        8.12
Jack        Lommi        4.50     9
Peter       Smith        2        5.75
...
```

Write a program that reads each line of students.txt and stores in `suc.txt` the names and grades of the students with average grade greater or equal to `5`, while in `fail.txt` the students with average grade less than `5`. The program should display the total number of students written in each file.

S.19 Suppose that the `test.bin` binary file contains a student's grades. Write a program that reads the grades (read them as **`float`**) from the binary file, then reads a number and displays those with a greater value than the input number.

S.20 What is the output of the following program?

```
#include <stdio.h>

#define no_main(type, name, text,num) type name() {printf(text);
return num;}

no_main(int, main, "No main() program", 0)
```

Indicative Answers

S.1 *Answer:*

The expression `a[i] = b[i]` is equivalent to `(a[i] = b[i]) != 0`, meaning that the elements of the array `b` would be copied to the respective elements of the array `a`, as long as the value of `a[i]` does not become `0`. If it does, the **`for`** loop terminates.

Since the type of the array `a` is **`int`**, only the integer parts of the `b` elements will be stored into `a`. Therefore, when the value `0.5` is copied, `a[2]` becomes `0` and the **`for`** loop terminates.

As a result, the program outputs: `2 1`.

S.2 *Answer:*

```
#include <stdio.h>

#define SIZE 50

int main()
{
	int i, j, num, found, code[SIZE];

	i = 0;
	while(i < SIZE)
	{
		printf("Enter code: ");
		scanf("%d", &num);

		found = 0;
		/* The variable i indicates how many codes have been
stored in the array. The for loop checks if the input code is
already stored. If it does, the variable found becomes 1 and the
loop terminates. */
		for(j = 0; j < i; j++)
		{
			if(code[j] == num)
			{
				printf("Error: Code %d exists. ", num);
				found = 1;
				break;
			}
```

```
            }
            /* If the code is not stored, we store it and the
index position is increased by one. */
            if(found == 0)
            {
                  code[i] = num;
                  i++;
            }
      }
      printf("\nCodes: ");
      for(i = 0; i < SIZE; i++)
            printf("%d ", code[i]);
      return 0;
}
```

S.3 ***Answer:***

With the statement `ptr2 = ptr1;` `ptr2` points to the same address with `ptr1`, that is the address of `i`. Therefore, `*ptr2` is equal to `i`.

Since both pointers point to the address of `i`, the statement `*ptr1 = *ptr1 + *ptr2;` is equivalent to `i = i+i = 10+10 = 20`.

Similarly, the statement `*ptr2 = 2*(*ptr2);` is equivalent to `i = 2*i = 2*20 = 40`. The program displays the value of the expression `*ptr1 + *ptr2;` that is `i+i = 40+40 = 80`.

S.4 ***Answer:***

```
#include <stdio.h>
int main()
{
      int *p1, *p2, *p3, i, j, sum;

      p1 = &i;
      p2 = &j;
      p3 = &sum;
      *p3 = 0;
      do
      {
            printf("Enter two numbers (a < b < 100): ");
            scanf("%d%d", p1, p2);
      } while(*p1 >= *p2 || *p2 > 100);

      (*p1)++;
      while(*p1 < *p2)
```

```
	{
		*p3 += *p1;
		(*p1)++;
	}
	printf("Sum = %d\n", *p3);
	return 0;
}
```

S.5 *Answer:*

```
#include <stdio.h>

#define SIZE 10

int main()
{
	int i, max_pos, min_pos;
	float max, min, arr[SIZE];

	max = 0;
	min = 10;
	max_pos = min_pos = 0;
	for(i = 0; i < SIZE; i++)
	{
		do
		{
			printf("Enter grade: ");
			scanf("%f", arr+i);
		} while(*(arr+i) > 10 || *(arr+i) < 0); /* Check if
the grade is within [0,10]. */
		if(*(arr+i) > max)
		{
			max = *(arr+i);
			max_pos = i;
		}
		if(*(arr+i) < min)
		{
			min = *(arr+i);
			min_pos = i;
		}
	}
	printf("Max grade is %.2f in pos #%d\n", max, max_pos);
	printf("Min grade is %.2f in pos #%d\n", min, min_pos);
	return 0;
}
```

S.6 *Answer:*

When the array `arr` is declared, `arr[0]` becomes `20` and the rest elements `0`.

With the statement `ptr = arr+1;` `ptr` points to `arr[1]`. In each loop iteration, the statement `ptr++` makes it point to the next element. The **`for`** loop is executed until the `ptr` points to the address of the last element.

In each loop iteration, the value of the current element becomes equal to the value of the previous element, plus the value of the next element, plus one. For example, in the first iteration, the statement:

`*ptr = *(ptr-1)+*(ptr+1)+1;` is equivalent to:
`arr[1] = arr[0]+arr[2]+1 = 20+0+1 = 21;`

As a result, the values of `arr[0]`, ... `arr[3]` become equal from `20` to `23`. What about the value of the last element?

`arr[4]` becomes equal to `arr[3]`, plus one, plus the random value that exists in the four-byte memory block following the address of `arr[4]`.

S.7 *Answer:*

```
#include <stdio.h>
int main()
{
	char ch, end_ch;

	end_ch = 'z';
	for(ch = 'a'; ch <= end_ch; ch++)
	{
		printf("%c ", ch);
		if(ch == 'z')
		{
			ch = 'A'-1; /* Subtract 1, so that the ch++ statement in the next loop iteration will make it equal to 'A'. */
			end_ch = 'Z'; /* Change the end character, so that the loop displays all upper case letters. */
			printf("\n");
		}
		else if(ch == 'Z')
		{
			ch = '0'-1;
			end_ch = '9';
			printf("\n");
		}
	}
	return 0;
}
```

S.8 *Answer:*

The expression `str1 == str2` compares `str1` and `str2` as pointers, not if they have the same content. Since `str1` and `str2` are stored in different memory addresses the program displays `Two`.

What would be the output if we write:

```
(*str1 == *str2) ? printf("One\n") : printf("Two\n");
```

Since `str1` can be used as a pointer to its first element, `*str1` is equal to `'t'`. Similarly, `*str2` is equal to `'t'`. Therefore, the program would display `One`.

S.9 *Answer:*

```
#include <stdio.h>
#include <string.h>
int main()
{
	char str[100];
	int i, ch;

	printf("Enter text (> 5 && < 100): ");
	while(1)
	{
		i = 0;
		while((ch = getchar()) != '\n' && ch != EOF)
		{
			if(i < 99)
			{
				str[i] = ch;
				i++;
			}
		}
		str[i] = '\0';
		if(strlen(str) > 5)
			break;
		else
			printf("Enter text (> 5 && < 100): ");
	}
	printf("%s\n", str);
	return 0;
}
```

S.10 *Answer:*

The first `strcpy()` copies the characters of the string `n`, that is the characters `'n'` and `'\0'` to `str[0]` and `str[1]` respectively, and returns the `str` pointer. Therefore, the expression `*strcpy(str, "n")` can be replaced by `*str`, that is `'n'`.

The second `strcpy()` copies the string `"xt"` in the third position of `str` and returns the `str+2` pointer. Therefore, the expression `*strcpy(str+2, "xt")` can be replaced by `*(str+2)`, that is `'x'`.

As a result, the program displays the product of the ASCII codes of `'n'` and `'x'`, that is `13200`. However, it doesn't display `next` as you might expect, but only `n`, because the first `strcpy()` replaced `'e'` with `'\0'`.

S.11 *Answer:*

```
#include <stdio.h>

double *max(double *ptr1, double *ptr2);

int main()
{
	double *ptr, i, j;

	printf("Enter numbers: ");
	scanf("%lf%lf", &i, &j);

	ptr = max(&i, &j);
	printf("The max of %f and %f is %f\n", i, j, *ptr);
	return 0;
}

double *max(double *ptr1, double *ptr2)
{
	if(*ptr1 > *ptr2)
		return ptr1;
	else
		return ptr2;
}
```

Comments: `max()` compares the two numbers and returns the pointer to the greater one. This pointer is copied to `ptr` and `printf()` displays the greater number. Note that without declaring `ptr`, we could write:

```
printf("The max value of %f and %f is %f\n", i, j, *max(&i, &j));
```

S.12 ***Answer:***

Since `strlen()` returns `4`, the **`for`** loop will be executed four times. Let's trace the iterations:

1st iteration (`i = 0`). The value of `ptr[0]` is displayed, that is `'T'`.

2nd iteration (`i++ = 1`). Since `ptr` has been increased by one, `ptr` points to the second character of the string, that is `'e'`. Since we handle `ptr` as an array, `ptr[0]` is `'e'` and `ptr[1]` is `'x'`. Therefore, the program displays `'x'`.

3rd iteration (`i++ = 2`). Now, `ptr` points to the third character of the string, that is `'x'`. Therefore, `ptr[0]` is `'x'`, `ptr[1]` is `'t'` and `ptr[2]` is equal to `'\0'`. As a result, the program displays nothing.

4th iteration (`i++ = 3`). Now, `ptr` points to the fourth character of the string, that is `'t'`. Therefore, `ptr[0]` is `'t'`, `ptr[1]` is `'\0'`, and `ptr[2]` and `ptr[3]` are equal to the values that exist in the address past `str+4`. Therefore, the program displays a random character.
To sum up, the program displays: `Tx(space)(random character)`.

S.13 ***Answer:***

```
#include <stdio.h>

void stat_arr(float arr[], int size, float *min, float *max,
float *avg);

int main()
{
	int i;
	float min, max, avg, arr[100];

	for(i = 0; i < 100; i++)
	{
		printf("Enter price: ");
		scanf("%f", &arr[i]);
		if(arr[i] == -1)
			break;
	}
	if(i == 0)
		return 0;
```

```
      /* The variable i indicates the number of the elements
stored into the array. For example, if the user doesn't enter
the value -1, i would be equal to 100. */
      stat_arr(arr, i, &min, &max, &avg);
      printf("Max=%.2f Min=%.2f Avg=%.2f\n", max, min, avg);
      return 0;
}

void stat_arr(float arr[], int size, float *min, float *max,
float *avg)
{
      int i;
      float sum;

      sum = *min = *max = arr[0];
      for(i = 1; i < size; i++)
      {
            if(arr[i] > *max)
                  *max = arr[i];
            if(arr[i] < *min)
                  *min = arr[i];
            sum += arr[i];
      }
      *avg = sum/size;
}
```

Comments: Since the **`return`** statement returns one value at most, we have to pass pointers as additional arguments.

S.14 *Answer:*

That's a tricky one. Since the value of `ptr` and not its address is passed to `test()` any changes in the value of `arg` don't affect the value of `ptr`. Therefore, the program displays: `Val = 30`.

S.15 *Answer:*

Notice that `main()` can be also called recursively. In each call, the value of `a` is decremented by `1`. The program stops calling `main()` once its value becomes `0`. Therefore, the program displays: `4 3 2 1`.

S.16 *Answer:*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
int main(int argc, char *argv[])
{
      char *rvs_str;
      int i, j, len;

      if(argc == 1)
      {
            printf("Missing string arguments ...\n");
            exit(1);
      }
      for(i = 1; i < argc; i++)
      {
            len = strlen(argv[i]);

            rvs_str = (char *) malloc(len+1); /* Allocate one
extra place for the null character. */
            if(rvs_str == NULL)
            {
                  printf("Error: Not available memory\n");
                  exit(1);
            }
            for(j = 0; j < len; j++)
                  rvs_str[j] = argv[i][len-1-j]; /* The last
character is stored in position len-1. */

            rvs_str[j] = '\0'; /* Terminate the string. */
            printf("Reverse of %s is: %s\n", argv[i], rvs_str);
            free(rvs_str);
      }
      return 0;
}
```

S.17 ***Answer:***

```
#include <stdio.h>
#include <string.h>

#define MAX_CITIES  5

struct city
{
/* Assume that 30 characters are enough to hold the names. */
      char name[30];
      char country[30];
      int pop;
};

int main()
{
```

```
	int i,pop;
	char country[30];
	struct city cities[MAX_CITIES];

	for(i = 0; i < MAX_CITIES; i++)
	{
		printf("\n***** Enter city data:\n");

		printf("City name: ");
		scanf("%s", cities[i].name);

		printf("Country name: ");
		scanf("%s", cities[i].country);

		printf("Population: ");
		scanf("%d", &cities[i].pop);
	}
	printf("\nCountry to search and population: ");
	scanf("%s%d", country, &pop);

	for(i = 0; i < MAX_CITIES; i++)
	  if((cities[i].pop > pop) && (strcmp(cities[i].country,
country) == 0))
		printf("%s (%d)\n", cities[i].name, cities[i].pop);

	return 0;
}
```

S.18 ***Answer:***

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
	FILE *fp_in, *fp_suc, *fp_fail;
	char fnm[100], lnm[100];
	int suc_stud, fail_stud;
	double grd1, grd2;

	fp_in = fopen("students.txt", "r");
	if(fp_in == NULL)
	{
		printf("Error: File can't be loaded\n");
		exit(1);
	}
	fp_suc = fopen("suc.txt", "w");
	if(fp_suc == NULL)
	{
		printf("Error: File_1 can't be created\n");
		exit(1);
```

```
	}
	fp_fail = fopen("fail.txt", "w");
	if(fp_fail == NULL)
	{
		printf("Error: File_2 can't be created\n");
		exit(1);
	}
	suc_stud = fail_stud = 0;
	while(1)
	{
		if(fscanf(fp_in,"%s%s%lf%lf",  fnm,  lnm,  &grd1,
&grd2) != 4)
			break;
		if((grd1+grd2)/2 >= 5)
		{
			fprintf(fp_suc,"%s  %s  %f  %f\n",  fnm,  lnm,
grd1, grd2);
			suc_stud++;
		}
		else
		{
			fprintf(fp_fail,"%s  %s  %f  %f\n",  fnm,  lnm,
grd1, grd2);
			fail_stud++;
		}
	}
	printf("Failed: %d Succeeded: %d\n", fail_stud, suc_stud);
	fclose(fp_suc);
	fclose(fp_fail);
	fclose(fp_in);
	return 0;
}
```

S.19 ***Answer:***

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
	FILE *fp;
	int i, grd_num;
	float grd, *grd_arr;

	fp = fopen("test.bin", "rb");
	if(fp == NULL)
	{
		printf("Error: File can't be loaded\n");
		exit(1);
	}
	fseek(fp, 0, SEEK_END);
```

```
	grd_num = ftell(fp) / sizeof(float); /* Since the file
pointer is at the end of file, ftell() returns the size of the
file in bytes. Since each grade is stored as float, their
division calculates the number of grades stored in the file. */
	fseek(fp, 0, SEEK_SET);
	grd_arr = (float *) malloc(grd_num * sizeof(float)); /*
Allocate memory to store the grades. */
	if(grd_arr == NULL)
	{
		printf("Error: Not available memory\n");
		exit(1);
	}
	/* Read all grades and check if they are read
successfully. */
	if(fread(grd_arr, sizeof(float), grd_num, fp) == grd_num)
	{
		printf("Enter grade: ");
		scanf("%f", &grd);
		for(i = 0; i < grd_num; i++)
			if(grd_arr[i] > grd)
				printf("%f\n", grd_arr[i]);
	}
	else
		printf("Error: fread() failed\n");

	free(grd_arr);
	fclose(fp);
	return 0;
}
```

S.20 *Answer:*

Here is a weird program with no `main()` included. The preprocessor substitutes the `type` with **`int`**, `name` with `main`, and so forth. Therefore, the preprocessor expands the `no_main` macro to:

```
int main() {printf("No main() program\n"); return 0;}
```

and the program displays: `No main() program`

Note that if we were using **`void`** instead of **`int`**, the compilation would fail, because a **`void`** function can't return a value.