



Programmering Grundkurs(DT106G)

EXAM

Exam Number: 41
Teacher: Uwe Köckemann
Contact Zoom Link: <https://oru-se.zoom.us/j/9230356989>
Date: 2021-01-14
Time: 14:15-19:15
Material: Open book
No. exercises: 5
Total points: 40
 ≥ 20 required to pass with grade 3
 ≥ 30 required for grade 4
 ≥ 35 required for grade 5
No. pages: 8 (excluding this page)

- Start each exercise on a **new page**.
- **Do not write your name or any other identifying information** anywhere, use your **exam number** only.
- If a question or exercise is unclear, make reasonable assumptions and **write them down**.
- If you write by hand, please use **clear handwriting** and **do not** use a **red pen**.
- Answers must be given in **English or Swedish**.
- Please use **short and complete sentences**.
- This is an open-book exam but you must write your own answers. Material copied from external sources is not permitted.

Helpful Notes

Operator Priority and Associativity

Priority	Category	Operator	Associativity
Highest	Unary postfix	<code>()</code> , <code>[]</code> , <code>-></code> , <code>.</code> , <code>++</code> , <code>--</code>	left
	Unary prefix	<code>!</code> , <code>++</code> , <code>--</code> , <code>+</code> , <code>-</code> , <code>*</code> , <code>&</code> , <code>sizeof</code> , <code>(type)</code>	right
	Multiplicative	<code>*</code> , <code>/</code> , <code>%</code>	left
	Addition etc.	<code>+</code> , <code>-</code>	left
	Comparison	<code><</code> , <code><=</code> , <code>>=</code> , <code>></code>	left
	Equality	<code>==</code> , <code>!=</code>	left
	Logic AND	<code>&&</code>	left
	Logic OR	<code> </code>	left
Lowest	Assign	<code>=</code> , <code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> , <code>%=</code>	right

stdlib.h

```
void *malloc(size_t size);
void *realloc(void *ptr, size_t size);
void free(void *ptr);
void exit(int status);
int rand(void);
void srand(unsigned int seed);
```

stdio.h

```
FILE *fopen(const char *path, const char *mode);
int fclose(FILE *stream);
int getc(FILE *stream);
int getchar(void);
int ungetc(int c, FILE *stream);
char *fgets(char *s, int size, FILE *stream);
int putc(int c, FILE *stream);
int printf(const char *format, ...);
int fprintf(FILE *stream, const char *format, ...);
int fscanf(FILE *stream, const char *format, ...);
int sscanf(const char *str, const char *format, ...);
size_t fread(void *ptr, size_t size, size_t nmemb,
             FILE *stream);
size_t fwrite(const void *ptr, size_t size, size_t nmemb,
             FILE *stream);
int fseek(FILE *stream, long int offset, int whence);
// whence can take either of these values:
// SEEK_SET, SEEK_CUR, or SEEK_END
long int ftell(FILE *stream);
```

```
int atoi(const char *str);  
// converts the string argument str to an integer
```

string.h

```
size_t strlen(const char *s);  
char *strcpy(char *dest, const char *src);  
char *strncpy(char *dest, const char *src, size_t n);  
int strcmp(const char *s1, const char *s2);  
int strncmp(const char *s1, const char *s2, size_t n);  
char *strcat(char *dest, const char *src);  
char *strncat(char *dest, const char *src, size_t n);  
char *strstr(const char *haystack, const char *needle);  
void *memmove(void *dest, const void *src, size_t n);
```

ctype.h

```
int isalnum(int c);  
int isalpha(int c);  
int isblank(int c);  
int isdigit(int c);  
int islower(int c);  
int isprint(int c);  
int ispunct(int c);  
int isspace(int c);  
int isupper(int c);  
int tolower(int c);  
int toupper(int c);
```

math.h

```
double sqrt(double x);  
double pow(double x, double y);  
double fabs(double x);  
double exp(double x);  
//returns the value of e raised to the xth power
```

Exercise 1 (8 points)

What will be the output of the code in each of the following code pieces? For each part you must motivate your answer by explaining why your answer is correct.

a)

```
#include <stdio.h>
int main() {
    int x;
    double y;
    x = 10 / 4 + 1 / 4;
    y = 10 / 4 + 1.0 / 4;
    printf("%d %lf", x, y);
}
```

b)

```
#include <stdio.h>
int main() {
    int i = -4;
    while(1) {
        if(i > 2)
            break;
        printf("%d ", ++i);
    }
}
```

c)

```
#include <stdio.h>
int main() {
    char x = 7;
    //in binary format 7 is 00000111
    printf("%d ", x << 2);
}
```

d)

```
#include <stdio.h>
void f1(int x)
{
    x = 10;
}
void f2(int *x)
{
    *x = 10 * x;
}
int main()
{
    int x = 5, y = 2;
    f1(x);
    f2(&y);
    printf("%d %d", x, y);
    return 0;
}
```

Exercise 2 (8 points)

```
#include <stdio.h>
#include <math.h>
int main()
{
    double x, y;
    printf("Enter x and y:");
    scanf("%lf%lf", &x, &y);

    ***

    return 0;
}
```

In the program above, 2 variables x and y are read from input. Complete the program by writing code that replaces `***` so that the program prints out the following mathematical expressions:

(a) $\frac{1+y}{2^{-x}}$

(b) $\sqrt{x} + \sqrt{y} + x^2 + y^2$

(c) e^{x-y}

(d) $|x^2 - y^2|$

Exercise 3 (10 points)

Write a program that calculates the standard deviation for a set of given numbers.

The standard deviation of n numbers (x_1, \dots, x_n) is calculated as follows:

$$sd = \sqrt{\frac{\sum_{i=1}^{i=n} (x_i - M)^2}{n}}$$

which is the same as:

$$sd = \sqrt{\frac{(x_1 - M)^2 + (x_2 - M)^2 + \dots + (x_{n-1} - M)^2 + (x_n - M)^2}{n}}$$

M is the mean (average) of the n numbers x_i and calculated as follows:

$$M = \frac{\sum_{i=1}^{i=n} x_i}{n}$$

which is the same as:

$$M = \frac{x_1 + x_2 + \dots + x_{n-1} + x_n}{n}$$

The program must work for any number of numbers (so any value of n). This means the program must ask the user to enter the number of numbers first, and then asks to enter n numbers. Thus you **cannot** use a static array (such as `double numbers[10]`) to hold the numbers. Instead, you must use a dynamic array (such as `double* numbers`) and allocate enough memory for it.

Exercise 4 (4 points)

Explain what is wrong with the following program and modify the code so that it becomes correct.

```
#include <stdio.h>
int main()
{
    int* x;
    scanf("%d", x);
    *x = *x * 10;
    printf("%d", *x);
    return 0;
}
```


Exercise 5 (10 points)

Suppose you have a binary file called books.dat that contains information about unknown number of books of the following type:

```
struct Book
{
    char title[120];
    char category[70];
    int edition;
    int publish_year;
    double price;
};
```

You are required to write a program that reads the contents of the file and writes the books into 3 new binary files with the following conditions:

1. The books that have the category “Mathematics” and are published after 2015 are written into the first file,
2. The books that have the category “Mathematics” and are published before 2015 are written into the second file,
3. The rest of the books (the books that do not fulfil the conditions in 1 or 2) must be written into the third file.