Projekt PP
Kiryl Alishkevich

a) W tym zadaniu wyprowadzam wszystkie niezbędne dane o bieżącym elemencie, a następnie zmieniam je na następny element za pomocą przechowywanego w nim wskaźnika.
b) Najpierw zapisuję rok wydania pierwszej książki, a następnie przechodzę przez wszystkie pozostałe, porównując ich rok wydania. Jeśli lata się zgadzają, zwiększam licznik i dodaję cenę do odpowiedniej zmiennej. Jeśli rok bieżącej książki jest mniejszy (książka jest starsza), aktualizuję wartości poprzednich zmiennych i zapisuję nowy "rekord".
c) Pytam o imie autora, a następnie przechodzę przez każdego autora każdej książki. Jeśli autor pasuje do wprowadzonego, zapisuję go w pliku.
d) Tworzę dynamiczną tablicę do przechowywania wszystkich lat wydania książek. Zapisuję w niej rok wydania pierwszej książki. Następnie przechodzę pętlą przez każdą książkę i sprawdzam jej rok wydania, jeśli nie ma jej jeszcze w tablicy, alokuję dla niej pamięć i zapisuję ją w tablicy. Sortuję tablicę za pomocą wbudowanej metody qsort, a następnie odwracam ją za pomocą funkcji, którą napisałem. Pętlą przechodzę przez tablicę z latami. Dla każdego roku zapisuję w pliku książki opublikowane w tym roku.
e) Tworzę tablicę dynamiczną z ciągami znaków, zapisuję w niej pierwszego autora. Przechodzę pętlą przez każdą książkę i jeśli jej autora nie ma w tablicy, alokuję mu miejsce i zapisuję go. W osobnej pętli przechodzę przez

każdego autora z tablicy, dla każdego autora sprawdzam, jakich książek jest autorem.

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

struct Book {
   char authors[3][100];
   char title[100];
   unsigned int price;
   unsigned int year;
};

struct Node {
   struct Book value;
   struct Node *next;
};

struct Node *getLast(struct Node *head) {
   if (head == NULL) {
       return NULL;
   }
   while (head->next) {
       head = head->next;
   }
   return head;
}

void pushBack(struct Node **head, struct Book data) {
   struct Node *last = getLast(*head);
   struct Node *tmp = (struct Node *) malloc(sizeof(struct
Node));
   tmp->value = data;
   tmp->next = NULL;
   last->next = tmp;
}


void printLinkedList(FILE *file, struct Node *head) {
   while (head) {
       fprintf(file, "%s %d %d %s %s %s\n",
               head->value.title,
               head->value.year,
               head->value.price,
```

```
                head->value.authors[0],
                head->value.authors[1],
                head->value.authors[2]);
        head = head->next;
    }
}

char **parseStringToArr(char input[]) {
    char **res = malloc(sizeof(char *) * 6);
    for (int i = 0; i < 6; i++)
        res[i] = malloc(100 * sizeof(char));

    char *token = strtok(input, " ");

    int i = 0;
    while (token != NULL) {
        strcpy(res[i], token);
        token = strtok(NULL, " ");
        i++;
    }

    if (i == 4) {
        res[4] = "\0";
        res[5] = "\0";
    } else if (i == 5) {
        res[5] = "\0";
    }

    return res;
}

void parseArrToBook(char **bookString, struct Book *book) {

    strcpy(book->title, bookString[0]);
    book->year = atoi(bookString[1]);
    book->price = atoi(bookString[2]);

    char tmp[200];
    for (int i = 3; i < 6; i++) {
        strcpy(tmp, bookString[i]);
        for (int j = 0; j < 200; j++) {
            if (tmp[j] == '\n') {
                tmp[j] = '\0';
            }
```

```
        }
        strcpy(book->authors[i - 3], tmp);
    }
}

int aTask(FILE *booksFile, struct Node head) {
    FILE *aTask = fopen("a.txt", "wt");

    if (booksFile == NULL) {
        printf("An error has occurred during opening file
\"a.txt\"");

        return 1;
    }

    printLinkedList(aTask, &head);

    fclose(aTask);

    return 0;
}

int bTask(FILE *booksFile, struct Node *head) {
    FILE *bTask = fopen("b.txt", "wt");

    if (booksFile == NULL) {
        printf("An error has occurred during opening file
\"b.txt\"");

        return 1;
    }

    unsigned int lowestYear = head->value.year;
    unsigned int oldestBookPrice = head->value.price;
    unsigned int priceArithmeticMean = head->value.price;
    int count = 1;
    while (1) {
        if (!head->next) {
            break;
        }
        head = head->next;
        if (head->value.year < lowestYear) {
            lowestYear = head->value.year;
            priceArithmeticMean = head->value.price;
```

```
            oldestBookPrice = head->value.price;
            count = 1;
        } else if (head->value.year == lowestYear) {
            count++;
            priceArithmeticMean += head->value.price;
        }
    }

    if (count > 1) {
        fprintf(bTask, "There were several oldest books. Price
arithmetic mean: %d", priceArithmeticMean / count);
    } else {
        fprintf(bTask, "There was only 1 oldest book. The price
is: %d", oldestBookPrice);
    }


    fclose(bTask);

    return 0;
}

int cTask(FILE *booksFile, struct Node *head) {
    FILE *cTask = fopen("c.txt", "wt");

    if (booksFile == NULL) {
        printf("An error has occurred during opening file
\"c.txt\"");

        return 1;
    }

    printf("Enter author name (f.e. 1BookAuthor1): \n");
    char author[100];
    scanf("%s", author);

    while (1) {
        for (int i = 0; i < 3; i++) {
            if (strcmp(head->value.authors[i], author) == 0) {
                fprintf(cTask, "%s \n", head->value.title);
                break;
            }
        }
        if (!head->next) {
```

```
            break;
        }
        head = head->next;
    }

    fclose(cTask);

    return 0;
}

int compare(const void *a, const void *b) {
    int int_a = *((int *) a);
    int int_b = *((int *) b);

    if (int_a == int_b) return 0;
    else if (int_a < int_b) return -1;
    else return 1;
}

void reverseArray(int arr[], int start, int end) {
    int temp;
    while (start < end) {
        temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}

int dTask(FILE *booksFile, struct Node *head) {
    struct Node *headCopy = head;

    FILE *dTask = fopen("d.txt", "wt");

    if (booksFile == NULL) {
        printf("An error has occurred during opening file
\"d.txt\"");

        return 1;
    }

    int *years;
    int size = 1;
```

```
    years = calloc(size, sizeof(int));

    int isThere = 0;
    years[0] = head->value.year;
    while (1) {
        for (int i = 0; i < size; i++) {
            if (years[i] == head->value.year) {
                isThere = 1;
                break;
            }
        }
        if (!isThere) {
            size++;
            years = realloc(years, size * sizeof(int));
            years[size - 1] = head->value.year;
        }

        isThere = 0;
        if (!head->next) {
            break;
        }
        head = head->next;
    }

    qsort(years, size, sizeof(int), compare);
    reverseArray(years, 0, size - 1);

    for (int i = 0; i < size; i++) {
        fprintf(dTask, "%d: \n", years[i]);
        head = headCopy;
        while (1) {
            if (head->value.year == years[i]) {
                fprintf(dTask, "\t%s %s\n", head->value.title,
head->value.authors[0]);
            }
            if (!head->next) {
                break;
            }
            head = head->next;
        }
    }

    fclose(dTask);
```

```
    return 0;
}

int eTask(FILE *booksFile, struct Node *head) {
    struct Node *headCopy = head;

    FILE *eTask = fopen("e.txt", "wt");

    if (booksFile == NULL) {
        printf("An error has occurred during opening file
\"e.txt\"");

        return 1;
    }

    int size = 1;
    char **authors;
    authors = calloc(1, sizeof(char *));
    authors[0] = malloc(100 * sizeof(char));
    strcpy(authors[size - 1], head->value.authors[0]);

    int isThere;
    while (1) {
        for (int i = 0; i < 3; i++) {
            isThere = 0;
            for (int j = 0; j < size; j++) {
                if (strcmp(head->value.authors[i], authors[j])
== 0) {
                    isThere = 1;
                    break;
                }
            }
            if (!isThere && (strcmp(head->value.authors[i], "")
!= 0)) {
                size++;
                authors = realloc(authors, size * sizeof(char
*));
                authors[size - 1] = malloc(100 * sizeof(char));
                strcpy(authors[size - 1],
head->value.authors[i]);
            }
        }

        if (!head->next) {
```

```
            break;
        }
        head = head->next;
    }

    for (int i = 0; i < size; i++) {
        fprintf(eTask, "%s: \n", authors[i]);
        head = headCopy;

        while (1) {
            for (int j = 0; j < 3; j++) {
                if (strcmp(head->value.authors[j], authors[i])
== 0) {
                    fprintf(eTask, "\t%s\n",
head->value.title);
                    break;
                }
            }

            if (!head->next) {
                break;
            }
            head = head->next;
        }
    }

    fclose(eTask);

    return 0;
}

int main() {
    FILE *booksFile = fopen("books.txt", "rt");
    if (booksFile == NULL) {
        printf("An error has occurred during opening file
\"books.txt\"");

        return 1;
    }

    char s[900];
    struct Book book;
    int i = 0;
```

```
    fgets(s, 1000, booksFile);
    struct Node *head = (struct Node *) malloc(sizeof(struct
Node));
    parseArrToBook(parseStringToArr(s), &book);
    head->value = book;
    head->next = NULL;


    while (fgets(s, 1000, booksFile) != NULL) {
        parseArrToBook(parseStringToArr(s), &book);
        pushBack(&head, book);
        i++;
    }

    if (aTask(booksFile, *head) == 1) return 2;
    if (bTask(booksFile, head) == 1) return 2;
    if (cTask(booksFile, head) == 1) return 2;
    if (dTask(booksFile, head) == 1) return 2;
    if (eTask(booksFile, head) == 1) return 2;

    fclose(booksFile);

    return 0;
}
```