Semestr: I
Projekt: 7.Biblioteka
Jezyk: C

7. Biblioteka
Program umożliwia zarządzanie biblioteką. Książki w bazie powinny być posortowane alfabetycznie wg. nazwiska autora. Baza danych powinna zawierać następujące informacje o książce: nr katalogowy, autor; tytuł; kategorię, datę wydania; wydawnictwo, nr ISBN, dostępność. Istnieje możliwość przeglądania listy posortowanej względem autora, tytułu oraz dostępności. Program powinien umożliwiać kasowanie książek według autora lub roku wydania. Nie można usunąć książki, jeżeli istnieje w bazie klient, który ją wypożyczył. Dodatkowo biblioteka powinna umożliwiać przechowywanie danych klientów: imię i nazwisko, adres, telefon oraz listę wypożyczonych przez klienta książek. Program umożliwia przeglądania klientów (wszystkich, wg początku nazwiska), ich dodawania, usuwanie i edycję. Klientów możemy przeglądać posortowanych w kolejności alfabetycznej lub wg liczby wypożyczonych pozycji.

Projekt zakłada stworzenie programu do zarządzania bazą danych biblioteki, umożliwia zarządzanie bazą czytelników oraz księgozbiorem. Jest realizowany przy użyciu list jednokierunkowych oraz plików.

Pliki użyte w programie:
Program przechowuje dane w postaci 2 plików tekstowych, w których dane są oddzielone znakami białymi, a każdy klient/książka znajduje się w osobnej linijce. Dodatkowo w pliku z czytelnikami na końcu każdej linijki znajduje się napis specjalny „@@@” który wskazuje na koniec danych danego klienta.

Listy użyte w programie:
Lista: lista_czytelnikow: znajdują się w niej dane na temat każdego czytelnika w tym jego lista książek w postaci listy lista_wypozyczonych
Lista: książek: lista_ksiazek zawiera dane na temat wszystkich ksiażek

Funkcje programu:

Odczytywanie: funkcja odczytuje dane z pliku z każdej linijki tworzy oddzielną strukturę z pliku książki.txt odczytuje konkretne liczbę kolumn a z pliku czytelnicy.txt odczytuje dane do znaku specjalnego „@@@”. W wyniku tej funkcji tworzone są podane wyżej listy.

Zapisywanie: funkcja zapisuje zmodyfikowanie dane do pliku.

Wypisanie: funkcja wypisuje na ekranie posortowane (w sposób wybrany przez użytkownika) dane.

Sortowanie:
Sortowanie list odbywa się poprzez przechodzenie przez listę dopóki nie będzie ona posortowana, funkcja porównuje po dwa elementy i ewentualnie zmienia wartości pomiędzy dwoma elementami listy.

W przypadku sortowania alfabetycznego wyrazy są porównywane za pomocą funkcji „strcmp”, a w przypadku liczb przez znak „ < ” . w przypadku kiedy nazwiska są takie same porównuje również imiona klientów lub autorów książek.

Funkcje związane z bazą danych klientów:
-Dodawanie klienta: funkcja dodaje nowego klienta do listy na ostatnie jej miejsce
-Usuwanie klienta: funkcja usuwa klienta o podanych przez użytkownika: imieniu i nazwisku
-Edycja klienta: funkcja wyszykuje klienta o podanych: imieniu i nazwisku a następnie umożliwia edycję jego danych

Usuwanie książek z bazy:
-Program umożliwia usuwanie wszystkich książek o zadanym parametrze (rok wydania lub autor) z bazy danych.

Program obsługuje się z poziomu konsoli za pomocą klawiszy cyfr oraz „/”, a także entera
Aby wybrać żądaną opcję należy podać przy wybranej funkcji cyfrę i nacisnąć enter.
Program zamiast spacji używa znaku „ _” i należy się tej zasady bezwarunkowo trzymać.

//Kod programu:

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

```
#include <stdlib.h>
#include <string.h>
struct lista_wypozyczonych //lista ksiazek wyporzyczonych przez konkretna osobe kazdy ma swoja liste
{
   char ksiazka[20];
   struct lista_wypozyczonych *nast1;
};
struct lista_ksiazek //lista ksiazek w bazie
{
   char imie[20], nazwisko[20], tytul[20], kategoria[20], wydawnictwo[20], dostepnosc[20], nr_katalogowy[20], nr_ISBN[20], data_wydania[5];
   int rok;
   //int nr_katalogowy, nr_ISBN, data_wydania;
   struct lista_ksiazek *nast;
};
struct lista_czytelnikow //lista czytelnikow w bazie
{
   char imie[20], nazwisko[20], adres[20], telefon[20];
   struct lista_wypozyczonych *wypozyczenia_osobiste;
   int ilosc_wypozyczonych;
   struct lista_czytelnikow *nast;

};
struct lista_czytelnikow* dodaj_klienta (struct lista_czytelnikow *czytelnicy) //dodanie nowego klienta do bazy jako ostatni element listy
{
   int i;
   struct lista_czytelnikow *tmp = czytelnicy;
   struct lista_wypozyczonych *tmp1, *tmp2=NULL;
   while(tmp->nast!=NULL)
   {
      tmp = tmp->nast;
   }
   tmp->nast = malloc(sizeof(struct lista_czytelnikow));
   tmp = tmp->nast;
   char wyraz[100];
   printf("Podaj: imie, nazwisko, adres, telefon oraz liste wypozyczonych ksiazek, zeby zakonczyc wpisywanie wpisz /\n");
   scanf("%s",tmp->imie);
   scanf("%s",tmp->nazwisko);
   scanf("%s",tmp->adres);
   scanf("%s",tmp->telefon);
   tmp1 = malloc(sizeof(struct lista_wypozyczonych));
   scanf("%s",wyraz);
```

```
   tmp->ilosc_wypozyczonych=0;
   i=(strcmp(wyraz,"/"));
   tmp2=tmp1;
   while(i!=0)
   {
      tmp->ilosc_wypozyczonych+=1;
      strcpy(tmp1->ksiazka, wyraz);
      tmp1->nast1 = malloc(sizeof(struct lista_wypozyczonych));
      tmp1 = tmp1->nast1;
      scanf("%s",wyraz);
      i=(strcmp(wyraz,"/"));
   }
tmp->wypozyczenia_osobiste=tmp2;
return czytelnicy;
}
struct lista_czytelnikow* edytuj_klienta (struct lista_czytelnikow *czytelnicy , char *imie ,
char *nazwisko) // edytownie danych klienta znalezionego po imieniu i nazwisku
{
   int i=1 , znak;
   struct lista_czytelnikow *tmp = czytelnicy;
   struct lista_wypozyczonych *tmp1, *tmp2=NULL;
   while(tmp!=NULL)
   {
      if ( strcmp(tmp->imie , imie)==0 && strcmp(tmp->nazwisko , nazwisko)==0 )
      {struct lista_wypozyczonych *tmp1;
         while(i==1)
         {
            printf("Obecne dane:\n");
      printf( "%s %s %s %s", tmp->imie, tmp->nazwisko, tmp->adres, tmp->telefon );
      tmp1 = tmp->wypozyczenia_osobiste;
      while(tmp1!=NULL)
      {
      printf( " %s", tmp1->ksiazka );
      tmp1 = tmp1->nast1;
      }
      printf("\n");
         printf("Co chcesz edytowac:\n1 Imie\n2 Nazwisko\n3 Adres\n4 Telefon\n5 Dodanie
ksiazki do listy wypozyczonych\n6 Usuniecie ksiazki z listy wypozyczonych\n7 zeby
zakonczyc edycje\n");
         scanf("%d", &znak);
         if (znak==1)
            {
              printf("Podaj nowe Imie: ");
              scanf("%s",tmp->imie);
            }
```

```
        else if (znak==2)
           {
              printf("Podaj nowe Nazwisko: ");
             scanf("%s",tmp->nazwisko);
           }
        else if (znak==3)
           {
             printf("Podaj nowy Adres: ");
             scanf("%s",tmp->adres);
           }
        else if (znak==4)
           {
              printf("Podaj nowy Telefon: ");
             scanf("%s",tmp->telefon);
           }
        else if (znak==5)
           {
             printf("Podaj tytul ksiazki ktora chcesz dodac: ");
             tmp1=tmp->wypozyczenia_osobiste;
             while(tmp1->nast1!=NULL) tmp1=tmp1->nast1;
             tmp1->nast1=malloc(sizeof(struct lista_wypozyczonych));
             tmp1=tmp1->nast1;
             tmp1->nast1=NULL;
             scanf("%s",tmp1->ksiazka);
             tmp->ilosc_wypozyczonych+=1;
           }
        else if (znak==6)
           {
                 char wyraz[100];
                 int czy=0;
                 printf("Podaj tytul ksiazki ktora chcesz usunac: ");
                 tmp1=tmp->wypozyczenia_osobiste;
                 scanf("%s",wyraz);
                 struct lista_wypozyczonych *tmp1=tmp->wypozyczenia_osobiste,
*poprzednia=tmp->wypozyczenia_osobiste;
                 if (strcmp(tmp1->ksiazka, wyraz)==0)
                 {
                    tmp->wypozyczenia_osobiste=tmp1->nast1;
                    free(tmp1);
                    czy=1;
                 }
                 tmp1 = poprzednia->nast1;
                 while(tmp1->nast1!=NULL)
                 {
                    if (strcmp(tmp1->ksiazka, wyraz)==0)
```

```
                    {
                        poprzednia->nast1=tmp1->nast1;
                        free(tmp1);
                        czy=1;
                    }
                    poprzednia=poprzednia->nast1;
                    tmp1 = poprzednia->nast1;
                }
                if (strcmp(tmp1->ksiazka, wyraz)==0)
                {
                    free(tmp1);
                    czy=1;
                    poprzednia->nast1=NULL;
                }
                if (czy=1) tmp->ilosc_wypozyczonych-=1;
            }
                else if (znak==7)
                {
                    i=0;
                }
                }
            }
            tmp=tmp->nast;
    }
return czytelnicy;
}
struct lista_ksiazek* odczyt_listy_ksiazek ( char *nazwa ) //odczytywanie listy ksiazek z pliku
{
    int i=0, j=0;
    char wyraz[100];
    struct lista_ksiazek *ksiazki2 = NULL;
    struct lista_ksiazek *ksiazki1, *tmp;
    FILE *plik;
    plik=fopen(nazwa,"r");
    ksiazki1=malloc(sizeof(struct lista_ksiazek));
    ksiazki1->nast=NULL;
    tmp=ksiazki2;
    while (fscanf (plik,"%s",&wyraz)!=EOF)
        {
                if (i==0) strcpy(ksiazki1->nr_katalogowy, wyraz);
                else if (i==1) strcpy(ksiazki1->imie, wyraz);
                else if (i==2) strcpy(ksiazki1->nazwisko, wyraz);
                else if (i==3) strcpy(ksiazki1->tytul, wyraz);
                else if (i==4) strcpy(ksiazki1->kategoria, wyraz);
                else if (i==5)
```

```
                {
                   strcpy(ksiazki1->data_wydania, wyraz);
                   sscanf(wyraz, "%d", &ksiazki1->rok);
                }
                else if (i==6) strcpy(ksiazki1->wydawnictwo, wyraz);
                else if (i==7) strcpy(ksiazki1->nr_ISBN, wyraz);
                else if (i==8) strcpy(ksiazki1->dostepnosc, wyraz);
                i++;
          if (i>=9 && j==0)
          {
             ksiazki2=ksiazki1;
             ksiazki1=malloc(sizeof(struct lista_ksiazek));
             ksiazki1->nast=NULL;
             j=1;
             i=0;
          }
          else if (i>=9)
          {
             i=0;
             tmp=ksiazki2;
             while(tmp->nast!=NULL)
             {
                tmp = tmp->nast;
             }
             tmp->nast = ksiazki1;
             ksiazki1=malloc(sizeof(struct lista_ksiazek));
             ksiazki1->nast=NULL;
          }
       }
   fclose(plik);
   return ksiazki2;
}
void zapis_listy_ksiazek ( struct lista_ksiazek* ksiazki, char *nazwa2 ) //zapisywanie nowej
listy ksiazek do pliku
{
   struct lista_ksiazek *tmp = ksiazki;
   FILE *plik;
   plik=fopen(nazwa2,"w");
   while (tmp!=NULL)
      {
         fprintf(plik,"%s %s %s %s %s %s %s %s %s\n", tmp->nr_katalogowy, tmp->imie,
tmp->nazwisko, tmp->tytul, tmp->kategoria, tmp->data_wydania, tmp->wydawnictwo, tmp-
>nr_ISBN, tmp->dostepnosc);
         tmp=tmp->nast;
      }
```

```
   fclose(plik);
}
struct lista_czytelnikow* odczyt_listy_czytelnikow (char *nazwa1) //odczytywanie listy
czytelnikow z pliku
{
   int i=0, j=0, k=0;
   char wyraz[20];
   struct lista_czytelnikow *czytelnicy2 = NULL;
   struct lista_czytelnikow *czytelnicy1, *tmp;
   struct lista_wypozyczonych *wypozyczenia_osobiste1 ;
   struct lista_wypozyczonych *wypozyczenia_osobiste2=NULL, *tmp1;
   wypozyczenia_osobiste1 = malloc(sizeof(struct lista_wypozyczonych));
   wypozyczenia_osobiste1->nast1=NULL ;
   FILE *plik;
   plik=fopen(nazwa1,"r");
   czytelnicy1=malloc(sizeof(struct lista_czytelnikow));
   czytelnicy1->nast=NULL;
   tmp=czytelnicy2;
   while (fscanf (plik,"%s",&wyraz)!=EOF)
      {
                  if (i==0) strcpy(czytelnicy1->imie, wyraz);
                  else if (i==1) strcpy(czytelnicy1->nazwisko, wyraz);
                  else if (i==2) strcpy(czytelnicy1->adres, wyraz);
                  else if (i==3) strcpy(czytelnicy1->telefon, wyraz);

                  if (i>=4 && strcmp(wyraz,"@@@")!=0 && k==0)
                  {
                     wypozyczenia_osobiste1=malloc(sizeof(struct lista_wypozyczonych));
                     wypozyczenia_osobiste1->nast1=NULL;
                     tmp1=wypozyczenia_osobiste2;
                     czytelnicy1->wypozyczenia_osobiste=wypozyczenia_osobiste1;
                     strcpy(wypozyczenia_osobiste1->ksiazka, wyraz);
                     k=1;
                     wypozyczenia_osobiste2=wypozyczenia_osobiste1;
                     wypozyczenia_osobiste1->nast1=NULL;
                     wypozyczenia_osobiste2->nast1=NULL;
                     }
                  else if (i>=5 && strcmp(wyraz,"@@@")!=0)
                  {
                     tmp1=wypozyczenia_osobiste2;
                     while(tmp1->nast1!=NULL)
                     {
                     tmp1 = tmp1->nast1;
                     }
                     wypozyczenia_osobiste1=malloc(sizeof(struct lista_wypozyczonych));
```

```
                    wypozyczenia_osobiste1->nast1=NULL;
                    strcpy(wypozyczenia_osobiste1->ksiazka, wyraz);
                    tmp1->nast1 = wypozyczenia_osobiste1;
                    wypozyczenia_osobiste1 = malloc(sizeof(struct lista_wypozyczonych));
                    wypozyczenia_osobiste1->nast1=NULL;
                }
                else if (strcmp(wyraz,"@@@")==0)
                {
                  if (j==0)
                   {
                      czytelnicy2=czytelnicy1;
                      czytelnicy1=malloc(sizeof(struct lista_czytelnikow));
                      czytelnicy1->nast=NULL;
                      czytelnicy2->ilosc_wypozyczonych=i-4;
                      j=1;
                      i=-1;
                      k=0;
                      struct lista_wypozyczonych *wypozyczenia_osobiste1 ;
                      struct lista_wypozyczonych *wypozyczenia_osobiste2=NULL, *tmp1;
                   }
                   else
                   {
                      tmp=czytelnicy2;
                      czytelnicy1->ilosc_wypozyczonych=i-4;
                      while(tmp->nast!=NULL) tmp = tmp->nast;
                         tmp->nast = czytelnicy1;
                      czytelnicy1=malloc(sizeof(struct lista_czytelnikow));
                      czytelnicy1->nast=NULL;
                      k=0;
                      i=-1;
                   }
                }
                i++;

       }
   fclose(plik);
   return czytelnicy2;
}
void zapis_listy_czytelnikow ( struct lista_czytelnikow* czytelnicy , char *nazwa3)
//zapisywanie nowej listy czytelnikow do pliku
{
   struct lista_czytelnikow *tmp = czytelnicy;
   struct lista_wypozyczonych *tmp1;
   FILE *plik;
   plik=fopen(nazwa3,"w");
```

```
   while (tmp!=NULL)
   {
      fprintf( plik , "%s %s %s %s" , tmp->imie, tmp->nazwisko, tmp->adres, tmp->telefon );
      tmp1 = tmp->wypozyczenia_osobiste;
      while(tmp1!=NULL)
      {
      fprintf( plik , " %s" , tmp1->ksiazka );
      tmp1 = tmp1->nast1;
      }
      tmp = tmp->nast;
      fprintf( plik , " @@@\n");
   }
   fclose( plik );
}
void wypisanie_listy_ksiazek (struct lista_ksiazek *ksiazki) //wypisywanie na ekran listy
ksiazek
{
   struct lista_ksiazek *tmp = ksiazki;
   printf("nr kat Imie Nazwisko Tytul Rodzaj Dat wyd Wydawnictwo nr ISBN status\n");
   while(tmp!=NULL)
   {
      printf("%s %s %s %s %s %s %s %s %s\n", tmp->nr_katalogowy, tmp->imie, tmp-
>nazwisko, tmp->tytul, tmp->kategoria, tmp->data_wydania, tmp->wydawnictwo, tmp-
>nr_ISBN, tmp->dostepnosc);
      tmp = tmp->nast;
   }
   printf("\n");
}
void wypisanie_listy_czytelnikow (struct lista_czytelnikow *czytelnicy) //wypisywanie na
ekran listy czytelnikow wraz z ich wypozyczeniami
{
   struct lista_czytelnikow *tmp = czytelnicy;
   struct lista_wypozyczonych *tmp1;
   printf ("Imie Nazwisko Adres nr Kom Lista wypozyczonych ksiazek\n");
   while(tmp!=NULL)
   {
      printf( "%s %s %s %s", tmp->imie, tmp->nazwisko, tmp->adres, tmp->telefon );
      tmp1 = tmp->wypozyczenia_osobiste;
      while(tmp1!=NULL)
      {
      printf( " %s", tmp1->ksiazka );
      tmp1 = tmp1->nast1;
      }
      tmp = tmp->nast;
      printf("\n");
```

```
    }
    printf("\n");
}
void wyszukaj_po_pocz_naz (struct lista_czytelnikow *czytelnicy, char *pocz_naz)
//wyszukiwanie osob ktorych poczatek nazwiska jest taki sam jak podany przez uzytkownika
{
    struct lista_czytelnikow *tmp = czytelnicy;
    struct lista_wypozyczonych *tmp1;
    int ilosc_liter=strlen(pocz_naz);
    while(tmp!=NULL)
    {
        if (strncmp(tmp->nazwisko,pocz_naz,ilosc_liter)==0)
        {
            printf( "%s %s %s %s", tmp->imie, tmp->nazwisko, tmp->adres, tmp->telefon );
            tmp1 = tmp->wypozyczenia_osobiste;
            while(tmp1!=NULL)
            {
            printf( " %s", tmp1->ksiazka );
            tmp1 = tmp1->nast1;
            }
            printf("\n");
        }
        tmp = tmp->nast;
    }
    printf("\n");
}
struct lista_ksiazek* usuniecie_ksiazek_autora (struct lista_ksiazek *ksiazki, char *imie, char
*nazwisko) //usuwanie wszytskich ksiazek danego autora z bazy o ile nie sa one
wypozyczone
{
    struct lista_ksiazek *tmp=ksiazki, *poprzednia=NULL, *tmp1=NULL;
    while ( tmp!=NULL && strcmp(tmp->dostepnosc,"dostepna")==0 && strcmp(tmp-
>imie,imie)==0 && strcmp(tmp->nazwisko,nazwisko)==0 )
    {
        tmp=tmp->nast;
    }
    ksiazki=tmp;
    poprzednia=ksiazki;
    if (ksiazki->nast==NULL) return ksiazki;
    tmp = poprzednia->nast;
    while(tmp!=NULL)
    {
        if ( (strcmp(tmp->imie,imie)!=0 && strcmp(tmp->nazwisko,nazwisko)!=0) ||
strcmp(tmp->dostepnosc , "niedostepna")==0 )
        {
```

```
            poprzednia=tmp;
        }
        else
        {
            poprzednia->nast=tmp->nast;
            tmp1=tmp;
            free(tmp1);
        }
        tmp = poprzednia->nast;
    }
return ksiazki;
}

struct lista_ksiazek* usuniecie_ksiazek_z_roku ( struct lista_ksiazek *ksiazki, int rok )
{
    struct lista_ksiazek *tmp=ksiazki, *poprzednia=NULL, *tmp1=NULL;
    while ( tmp!=NULL && strcmp(tmp->dostepnosc,"dostepna")==0 && tmp->rok==rok )
    {
        tmp=tmp->nast;
    }
    ksiazki=tmp;
    poprzednia=ksiazki;
    if (ksiazki->nast==NULL) return ksiazki;
    tmp = poprzednia->nast;
    while(tmp!=NULL)
    {
        if ( tmp->rok!=rok || strcmp(tmp->dostepnosc , "niedostepna")==0 )
        {
            poprzednia=tmp;
        }
        else
        {
            poprzednia->nast=tmp->nast;
            tmp1=tmp;
            free(tmp1);
        }
        tmp = poprzednia->nast;
    }
return ksiazki;
}

struct lista_czytelnikow* usuniecie_czytelnika (struct lista_czytelnikow *czytelnicy, char
*imie, char *nazwisko) //usuwanie czytelnika o podanym imieniu i nazwisku
{
    struct lista_czytelnikow *tmp=czytelnicy, *poprzedni=czytelnicy;
```

```
    if ( strcmp(tmp->imie , imie)==0 && strcmp(tmp->nazwisko , nazwisko)==0 )
    {
        struct lista_wypozyczonych *tmp1=czytelnicy->wypozyczenia_osobiste, *tmp2;
        while ( tmp1 != NULL)
        {
            tmp2=tmp1->nast1;
            free(tmp1);
            tmp1=tmp2;
        }
        czytelnicy=tmp->nast;
        free(tmp);
    }
    tmp = poprzedni->nast;
    while(tmp->nast!=NULL)
    {
        if ( strcmp(tmp->imie , imie)==0 && strcmp(tmp->nazwisko , nazwisko)==0 )
        {
            struct lista_wypozyczonych *tmp1=tmp->wypozyczenia_osobiste, *tmp2;
            while ( tmp1 != NULL)
            {
                tmp2=tmp1->nast1;
                free(tmp1);
                tmp1=tmp2;
            }
            poprzedni->nast=tmp->nast;
            free(tmp);
        }
        poprzedni=poprzedni->nast;
        tmp = poprzedni->nast;
    }
    if ( strcmp(tmp->imie , imie)==0 && strcmp(tmp->nazwisko , nazwisko)==0 )
    {
        struct lista_wypozyczonych *tmp1=tmp->wypozyczenia_osobiste, *tmp2;
        while ( tmp1 != NULL)
        {
            tmp2=tmp1->nast1;
            free(tmp1);
            tmp1=tmp2;
        }
        free(tmp);
        poprzedni->nast=NULL;
    }
    return czytelnicy;
}
```

```
struct lista_ksiazek* sortowanie_po_tytule (struct lista_ksiazek *ksiazki) //sortowanie
alfabetyczne ksiazek po ich tytule
{
   struct lista_ksiazek *obecna=ksiazki, *nastepna=NULL, *tmp=NULL, *tmp3=NULL;
   int i=0, liczba;
   char wyraz[100];
   nastepna=obecna->nast;
   while (i==0)
   {
      i=1;
      while (nastepna!=NULL)
      {//printf("%s # %s\n",obecna->tytul,nastepna->tytul);
         if (strcmp((obecna->tytul),(nastepna->tytul))>0)
         {
            strcpy(wyraz,obecna->nr_katalogowy);
            strcpy(obecna->nr_katalogowy,nastepna->nr_katalogowy);
            strcpy(nastepna->nr_katalogowy,wyraz);
            strcpy(wyraz,obecna->imie);
            strcpy(obecna->imie,nastepna->imie);
            strcpy(nastepna->imie,wyraz);
            strcpy(wyraz,obecna->nazwisko);
            strcpy(obecna->nazwisko,nastepna->nazwisko);
            strcpy(nastepna->nazwisko,wyraz);
            strcpy(wyraz,obecna->tytul);
            strcpy(obecna->tytul,nastepna->tytul);
            strcpy(nastepna->tytul,wyraz);
            strcpy(wyraz,obecna->kategoria);
            strcpy(obecna->kategoria,nastepna->kategoria);
            strcpy(nastepna->kategoria,wyraz);
            strcpy(wyraz,obecna->data_wydania);
            strcpy(obecna->data_wydania,nastepna->data_wydania);
            strcpy(nastepna->data_wydania,wyraz);
            liczba=obecna->rok;
            obecna->rok=nastepna->rok;
            nastepna->rok=liczba;
            strcpy(wyraz,obecna->wydawnictwo);
            strcpy(obecna->wydawnictwo,nastepna->wydawnictwo);
            strcpy(nastepna->wydawnictwo,wyraz);
            strcpy(wyraz,obecna->nr_ISBN);
            strcpy(obecna->nr_ISBN,nastepna->nr_ISBN);
            strcpy(nastepna->nr_ISBN,wyraz);
            strcpy(wyraz,obecna->dostepnosc);
            strcpy(obecna->dostepnosc,nastepna->dostepnosc);
            strcpy(nastepna->dostepnosc,wyraz);
            i=0;
```

```
            }
            if (tmp3==NULL)
            {
               tmp3=malloc(sizeof(struct lista_ksiazek));
               tmp3=obecna;
            }
            //printf("%s $ %s\n",obecna->tytul,nastepna->tytul);
            //wypisanie_listy_ksiazek (ksiazki);
            obecna=obecna->nast;
            nastepna=obecna->nast;
         }
         obecna=tmp3;
         nastepna=obecna->nast;
         //printf("%d\n",i);
      }
return tmp3;
}
struct lista_ksiazek* sortowanie_po_autorze (struct lista_ksiazek *ksiazki) //sortowanie alfabetyczne ksiazek po nazwisku ich autora, jesli imiona si powtarzaja posortowane sa takze po imieniu
{
   struct lista_ksiazek *obecna=ksiazki, *nastepna=NULL, *tmp=NULL, *tmp3=NULL;
   int i=0, liczba;
   char wyraz[100];
   nastepna=obecna->nast;
   while (i==0)
   {
      i=1;
      while (nastepna!=NULL)
      {//printf("%s # %s\n",obecna->tytul,nastepna->tytul);
         if (strcmp((obecna->nazwisko),(nastepna->nazwisko))>0)
         {
            strcpy(wyraz,obecna->nr_katalogowy);
            strcpy(obecna->nr_katalogowy,nastepna->nr_katalogowy);
            strcpy(nastepna->nr_katalogowy,wyraz);
            strcpy(wyraz,obecna->imie);
            strcpy(obecna->imie,nastepna->imie);
            strcpy(nastepna->imie,wyraz);
            strcpy(wyraz,obecna->nazwisko);
            strcpy(obecna->nazwisko,nastepna->nazwisko);
            strcpy(nastepna->nazwisko,wyraz);
            strcpy(wyraz,obecna->tytul);
            strcpy(obecna->tytul,nastepna->tytul);
            strcpy(nastepna->tytul,wyraz);
            strcpy(wyraz,obecna->kategoria);
```

```
    strcpy(obecna->kategoria,nastepna->kategoria);
    strcpy(nastepna->kategoria,wyraz);
    strcpy(wyraz,obecna->data_wydania);
    strcpy(obecna->data_wydania,nastepna->data_wydania);
    strcpy(nastepna->data_wydania,wyraz);
    liczba=obecna->rok;
    obecna->rok=nastepna->rok;
    nastepna->rok=liczba;
    strcpy(wyraz,obecna->wydawnictwo);
    strcpy(obecna->wydawnictwo,nastepna->wydawnictwo);
    strcpy(nastepna->wydawnictwo,wyraz);
    strcpy(wyraz,obecna->nr_ISBN);
    strcpy(obecna->nr_ISBN,nastepna->nr_ISBN);
    strcpy(nastepna->nr_ISBN,wyraz);
    strcpy(wyraz,obecna->dostepnosc);
    strcpy(obecna->dostepnosc,nastepna->dostepnosc);
    strcpy(nastepna->dostepnosc,wyraz);
    i=0;
}
else if (strcmp((obecna->nazwisko),(nastepna->nazwisko))==0)
{
    if (strcmp((obecna->nazwisko),(nastepna->nazwisko))>0)
    {
    strcpy(wyraz,obecna->nr_katalogowy);
    strcpy(obecna->nr_katalogowy,nastepna->nr_katalogowy);
    strcpy(nastepna->nr_katalogowy,wyraz);
    strcpy(wyraz,obecna->imie);
    strcpy(obecna->imie,nastepna->imie);
    strcpy(nastepna->imie,wyraz);
    strcpy(wyraz,obecna->nazwisko);
    strcpy(obecna->nazwisko,nastepna->nazwisko);
    strcpy(nastepna->nazwisko,wyraz);
    strcpy(wyraz,obecna->tytul);
    strcpy(obecna->tytul,nastepna->tytul);
    strcpy(nastepna->tytul,wyraz);
    strcpy(wyraz,obecna->kategoria);
    strcpy(obecna->kategoria,nastepna->kategoria);
    strcpy(nastepna->kategoria,wyraz);
    strcpy(wyraz,obecna->data_wydania);
    strcpy(obecna->data_wydania,nastepna->data_wydania);
    strcpy(nastepna->data_wydania,wyraz);
    liczba=obecna->rok;
    obecna->rok=nastepna->rok;
    nastepna->rok=liczba;
    strcpy(wyraz,obecna->wydawnictwo);
```

```
            strcpy(obecna->wydawnictwo,nastepna->wydawnictwo);
            strcpy(nastepna->wydawnictwo,wyraz);
            strcpy(wyraz,obecna->nr_ISBN);
            strcpy(obecna->nr_ISBN,nastepna->nr_ISBN);
            strcpy(nastepna->nr_ISBN,wyraz);
            strcpy(wyraz,obecna->dostepnosc);
            strcpy(obecna->dostepnosc,nastepna->dostepnosc);
            strcpy(nastepna->dostepnosc,wyraz);
            i=0;
            }
         }
         if (tmp3==NULL)
         {
            tmp3=malloc(sizeof(struct lista_ksiazek));
            tmp3=obecna;
         }
         //printf("%s $ %s\n",obecna->tytul,nastepna->tytul);
         //wypisanie_listy_ksiazek (ksiazki);
         obecna=obecna->nast;
         nastepna=obecna->nast;
      }
      obecna=tmp3;
      nastepna=obecna->nast;
      //printf("%d\n",i);
   }
return tmp3;
}
void wyswietlenie_posortowanej_po_dostepnosci (struct lista_ksiazek *ksiazki)
//wyswietlanie listy ksiazek w zaleznosci czy sa dostepne czy nie
{
   struct lista_ksiazek *tmp = ksiazki;
   while(tmp!=NULL)
   {
      if (strcmp((tmp->dostepnosc),"dostepna")==0)
         printf("%s %s %s %s %s %s %s %s %s\n", tmp->nr_katalogowy, tmp->imie, tmp-
>nazwisko, tmp->tytul, tmp->kategoria, tmp->data_wydania, tmp->wydawnictwo, tmp-
>nr_ISBN, tmp->dostepnosc);
      tmp = tmp->nast;
   }
   tmp = ksiazki;
   while(tmp!=NULL)
   {
      if (strcmp((tmp->dostepnosc),"niedostepna")==0)
```

```
        printf("%s %s %s %s %s %s %s %s %s\n", tmp->nr_katalogowy, tmp->imie, tmp->nazwisko, tmp->tytul, tmp->kategoria, tmp->data_wydania, tmp->wydawnictwo, tmp->nr_ISBN, tmp->dostepnosc);
        tmp = tmp->nast;
    }
    printf("\n");
}
struct lista_czytelnikow* sortowanie_po_nazwisku_i_imieniu (struct lista_czytelnikow *czytelnicy) //sortowanie listy czytelnikow po ich nazwisku i imieniu
{
    struct lista_czytelnikow *obecna=czytelnicy, *nastepna=NULL, *tmp=NULL, *tmp3=NULL;
    struct lista_wypozyczonych *wypozyczone=NULL;
    wypozyczone=malloc(sizeof(struct lista_wypozyczonych));
    int i=0, liczba;
    char wyraz[100];
    nastepna=obecna->nast;
    while (i==0)
    {
        i=1;
        while (nastepna!=NULL)
        {
            if (strcmp((obecna->nazwisko),(nastepna->nazwisko))>0)
            {
                strcpy(wyraz,obecna->adres);
                strcpy(obecna->adres,nastepna->adres);
                strcpy(nastepna->adres,wyraz);
                strcpy(wyraz,obecna->imie);
                strcpy(obecna->imie,nastepna->imie);
                strcpy(nastepna->imie,wyraz);
                strcpy(wyraz,obecna->nazwisko);
                strcpy(obecna->nazwisko,nastepna->nazwisko);
                strcpy(nastepna->nazwisko,wyraz);
                strcpy(wyraz,obecna->telefon);
                strcpy(obecna->telefon,nastepna->telefon);
                strcpy(nastepna->telefon,wyraz);
                liczba=obecna->ilosc_wypozyczonych;
                obecna->ilosc_wypozyczonych=nastepna->ilosc_wypozyczonych;
                nastepna->ilosc_wypozyczonych=liczba;
                wypozyczone=obecna->wypozyczenia_osobiste;
                obecna->wypozyczenia_osobiste=nastepna->wypozyczenia_osobiste;
                nastepna->wypozyczenia_osobiste=wypozyczone;
                i=0;
            }
            else if (strcmp((obecna->nazwisko),(nastepna->nazwisko))==0)
```

```
            {
               if (strcmp((obecna->nazwisko),(nastepna->nazwisko))>0)
               {
               strcpy(wyraz,obecna->adres);
               strcpy(obecna->adres,nastepna->adres);
               strcpy(nastepna->adres,wyraz);
               strcpy(wyraz,obecna->imie);
               strcpy(obecna->imie,nastepna->imie);
               strcpy(nastepna->imie,wyraz);
               strcpy(wyraz,obecna->nazwisko);
               strcpy(obecna->nazwisko,nastepna->nazwisko);
               strcpy(nastepna->nazwisko,wyraz);
               strcpy(wyraz,obecna->telefon);
               strcpy(obecna->telefon,nastepna->telefon);
               strcpy(nastepna->telefon,wyraz);
               liczba=obecna->ilosc_wypozyczonych;
               obecna->ilosc_wypozyczonych=nastepna->ilosc_wypozyczonych;
               nastepna->ilosc_wypozyczonych=liczba;
               wypozyczone=obecna->wypozyczenia_osobiste;
               obecna->wypozyczenia_osobiste=nastepna->wypozyczenia_osobiste;
               nastepna->wypozyczenia_osobiste=wypozyczone;
               i=0;
               }
            }
            if (tmp3==NULL)
            {
               tmp3=malloc(sizeof(struct lista_ksiazek));
               tmp3=obecna;
            }
            //printf("%s $ %s\n",obecna->tytul,nastepna->tytul);
            //wypisanie_listy_ksiazek (ksiazki);
            obecna=obecna->nast;
            nastepna=obecna->nast;
         }
         obecna=tmp3;
         nastepna=obecna->nast;
         //printf("%d\n",i);
      }
free(wypozyczone);
return tmp3;
}
struct lista_czytelnikow* sort_il_poz (struct lista_czytelnikow *czytelnicy) //sortowanie listy
czytelnikow w zaleznosci od ilosci wypozyczonych przez nich pozycji
{
```

```
   struct lista_czytelnikow *obecna=czytelnicy, *nastepna=NULL, *tmp=NULL,
*tmp3=NULL;
   struct lista_wypozyczonych *wypozyczone=NULL;
   wypozyczone=malloc(sizeof(struct lista_wypozyczonych));
   int i=0, liczba;
   char wyraz[100];
   nastepna=obecna->nast;
   while (i==0)
   {
      i=1;
      while (nastepna!=NULL)
      {
         if ((obecna->ilosc_wypozyczonych)<(nastepna->ilosc_wypozyczonych))
         {
            strcpy(wyraz,obecna->adres);
            strcpy(obecna->adres,nastepna->adres);
            strcpy(nastepna->adres,wyraz);
            strcpy(wyraz,obecna->imie);
            strcpy(obecna->imie,nastepna->imie);
            strcpy(nastepna->imie,wyraz);
            strcpy(wyraz,obecna->nazwisko);
            strcpy(obecna->nazwisko,nastepna->nazwisko);
            strcpy(nastepna->nazwisko,wyraz);
            strcpy(wyraz,obecna->telefon);
            strcpy(obecna->telefon,nastepna->telefon);
            strcpy(nastepna->telefon,wyraz);
            liczba=obecna->ilosc_wypozyczonych;
            obecna->ilosc_wypozyczonych=nastepna->ilosc_wypozyczonych;
            nastepna->ilosc_wypozyczonych=liczba;
            wypozyczone=obecna->wypozyczenia_osobiste;
            obecna->wypozyczenia_osobiste=nastepna->wypozyczenia_osobiste;
            nastepna->wypozyczenia_osobiste=wypozyczone;
            i=0;
         }
         if (tmp3==NULL)
         {
            tmp3=malloc(sizeof(struct lista_ksiazek));
            tmp3=obecna;
         }
         obecna=obecna->nast;
         nastepna=obecna->nast;
      }
      obecna=tmp3;
      nastepna=obecna->nast;
   }
```

```
free(wypozyczone);
return tmp3;
}
int main()
{
   char nazwa[17]="ksiazki.txt" , nazwa1[15]="Czytelnicy.txt" , nazwa2[13]="Ksiazki1.txt" ,
imie[100]="Jan" , nazwisko[100]="Kozak", rok[5]="2000" , nazwa3[16]="Czytelnicy1.txt";
   struct lista_ksiazek *ksiazki = NULL;
   int dzialanie=1, funkcja, funkcja1;
   struct lista_czytelnikow *czytelnicy = NULL;
   ksiazki = odczyt_listy_ksiazek (nazwa); //zaimpotowanie ksiazek z pliku do programu
   czytelnicy = odczyt_listy_czytelnikow (nazwa1); //zaimpotowanie danych klientow z pliku
do programu
   if (ksiazki==NULL || czytelnicy==NULL) return(0);
   /*wypisanie_listy_ksiazek (ksiazki);
   printf("////////////////////////////////////\n");
   ksiazki=sortowanie_po_tytule (ksiazki);
   wypisanie_listy_ksiazek (ksiazki);
   ksiazki=sortowanie_po_autorze (ksiazki);
   wypisanie_listy_ksiazek (ksiazki);
   ksiazki = odczyt_listy_ksiazek (nazwa);
   wypisanie_listy_ksiazek (ksiazki);
   czytelnicy = odczyt_listy_czytelnikow (nazwa1);
   zapis_listy_ksiazek ( ksiazki, nazwa2 );
   //dodanie_liczby_wypozyczonych (czytelnicy);
   wypisanie_listy_czytelnikow (czytelnicy);
   //ksiazki=usuniecie_ksiazek_autora (ksiazki, imie, nazwisko);
   //ksiazki=usuniecie_ksiazek_z_roku ( ksiazki , rok );
   wypisanie_listy_ksiazek (ksiazki);
   //czytelnicy=dodaj_klienta (czytelnicy);
   //czytelnicy=usuniecie_czytelnika1 ( czytelnicy , imie , nazwisko );
   czytelnicy=edytuj_klienta ( czytelnicy , imie , nazwisko );
   printf("////////////////////////////////////\n");
   zapis_listy_czytelnikow ( czytelnicy , nazwa3 );
   wypisanie_listy_czytelnikow (czytelnicy);*/
   while(dzialanie==1)
        {
         printf("Co chcesz zrobic:\n1 Wyswietlanie zbioru ksiazek\n2 Wyswietlanie listy
czytelnikow\n3 Usuwanie/Dodawanie/Edycja Czytelnikow\n4 Wyszukiwanie po poczatku
nazwiska \n5 Usuwanie ksiazek \n6 Zakonczenie dzialania programu\n");
         scanf("%d",&funkcja);
         system("cls");
         if (funkcja==1)
            {
```

```
                printf("Jaki rodzaj sortowania wybierasz:\n1 Po tytule\n2 Po autorze\n3 Po
dostepnosci\n");
                scanf("%d",&funkcja);
                if (funkcja==1)
                {
                   wypisanie_listy_ksiazek (sortowanie_po_tytule (ksiazki));
                }
                else if (funkcja==2)
                {
                   wypisanie_listy_ksiazek (sortowanie_po_autorze (ksiazki));
                }
                else if (funkcja==3)
                {
                   wyswietlenie_posortowanej_po_dostepnosci (ksiazki);
                }
             }
          else if (funkcja==2)
             {
                printf("Jaki rodzaj sortowania wybierasz:\n1 Po nazwisku i imieniu\n2 Po liczbie
wypozyczonych pozycji\n");
                scanf("%d",&funkcja);
                if (funkcja==1)
                {
                   wypisanie_listy_czytelnikow
(sortowanie_po_nazwisku_i_imieniu(czytelnicy));
                }
                else if (funkcja==2)
                {
                   wypisanie_listy_czytelnikow (sort_il_poz (czytelnicy));
                }
             }
          else if (funkcja==3)
             {
               printf("Co chcesz zrobic:\n1 Usuwanie czytelnika\n2 Dodawanie czytelnika\n3
Edycja czytelnika\n");
                scanf("%d",&funkcja);
                if (funkcja==1)
                {
                   printf("Kogo chcesz usunac, podaj imie i nazwisko:\n");
                   scanf("%s%s",imie,nazwisko);
                   czytelnicy=usuniecie_czytelnika ( czytelnicy , imie , nazwisko );
                }
                else if (funkcja==2)
                {
                   czytelnicy=dodaj_klienta (czytelnicy);
```

```
                }
                else if (funkcja==3)
                {
                    printf("Kogo chcesz edytowac, podaj imie i nazwisko:\n");
                    scanf("%s%s",imie,nazwisko);
                    czytelnicy=edytuj_klienta ( czytelnicy , imie , nazwisko );
                }
            }
        else if (funkcja==4)
            {
                char pocz_naz[50];
                printf("Podaj poczatek nazwiska: ");
                scanf("%s",pocz_naz);
                wyszukaj_po_pocz_naz (czytelnicy, pocz_naz);

            }
        else if (funkcja==5)
            {
                printf("Co chcesz zrobic:\n1 Usuwanie ksiazek danego autora\n2 Usuwanie
ksiazek z danego roku\n");
                scanf("%d",&funkcja);
                if (funkcja==1)
                {
                    printf("Jakiego autora chcesz usunac, podaj imie i nazwisko: \n");
                    scanf("%s%s",imie,nazwisko);
                    ksiazki=usuniecie_ksiazek_autora (ksiazki, imie, nazwisko);
                }
                if (funkcja==2)
                {
                    printf("Z jakiego roku chcesz usunac ksiazki: \n");
                    int rok;
                    scanf("%d",&rok);
                    ksiazki=usuniecie_ksiazek_z_roku ( ksiazki , rok );
                }
            }
        else if (funkcja==6)
            {
                dzialanie=0;
                zapis_listy_ksiazek ( ksiazki, nazwa2 ); //zapis zaktualizowanej listy ksiazek do
pliku
                zapis_listy_czytelnikow ( czytelnicy , nazwa3 ); //zapis zaktualizowanej listy
klientow do pliku
            }
        }
}
```