Podstawy Programowania
Zajęcia laboratoryjne 13
I rok Bioinformatyki Politechniki Poznańskiej

Operacje na plikach

1 Zapis/Odczyt do pliku - informacje podstawowe

Standardowe strumienie wejścia i wyjścia, wyróżniamy:

- stdin - strumień wejściowy (klawiatura),
- stdout - strumień wyjściowy (ekran),
- stderr - strumień komunikatów błędów (ekran).

Operacje na plikach, realizowane są za pomocą strumieni. Strumienie reprezentowane są poprzez zmienne typu FILE (taka struktura tworzona jest automatycznie podczas otwierania strumienia i zawiera informacje między innymi o nazwie pliku czy trybie otwarcia). Wszystkie kolejne operacje na pliku wymagają podania wskaźnika na tą strukturę, poniżej przykład:

```
1 FILE *fp; //definicja zmiennej plikowej
```

Funkcje realizujące operacje na plikach, zawarte są w bibliotece <stdio.h>, najczęściej używane funkcje to: fopen, fclose, fwrite, fread, fprintf, fscanf, feof. Przykłady z wyjaśnieniem poniżej:

```
1  //otwarcie pliku w odpowiednim trybie
2  FILE * fopen (char *nazwa_pliku, char *rodzaj_operacji);
3  /* Rodzaj operacji:
4  r - tylko do odczytu
5  w - tylko do zapisu
6  a - dopisywanie na koncu
7  r+ - z mozliwoscia aktualizacji
8  b - otwarcie jako plik binarny */
9
10 //zamkniecie pliku
11 int fclose (FILE *strumien);
12
13 //odczyt i zapis formatowany
14 int fscanf(FILE *strumien, char *format, ...); //analogicznie jak scanf
15 int fprintf(FILE *strumien, char *format, ...); //analogicznie jak printf
16
17 //odczyt i zapis calej linii lub lancucha znakow (do co najwyzej okreslonej dlugosci)
18 char* fgets(char *tekst, int dlugosc, FILE *strumien);
19 int fputs(char *tekst, FILE *strumien);
20
21 //odczyt i zapis pojedynczego znaku
22 int fgetc(FILE *strumien); //wczytanie pojedynczego znaku
23 int fputc(int znak, FILE *strumien); //wyslanie pojedynczego znaku
24
25 //odczyt i zapis pliku binarnego
26 int fread(void* adres, size_t rozm_bl, size_t il_blokow, FILE *strumien);
27 int fwrite(void* adres, size_t rozm_bl, size_t il_blokow, FILE *strumien);
```

```
28
29 //testowanie osiagniecia konca pliku
30 int feof(FILE *strumien);
31
32 //czyszczenie bufora
33 int fflush(FILE *strumien); //czysci bufor wskazanego strumienia
34 int flushall(); //czysci bufor dla wszystkich buforowanych strumieni
```

Przykładowy kod zapisu do pliku (kod 1 w materiałach na stronie):

```
1  #include <stdio.h>
2  #include<stdlib.h>
3
4  int main()
5  {
6      FILE *fp = fopen("test.txt", "w"); //identyfikator pliku
7      if(fp == NULL)
8      {
9          printf("blad otwierania pliku!\n");
10         return 0;
11     }
12     //zapisz do pliku "jakis tekst"
13     fprintf(fp, "Jakis tekst\n");
14
15     //zapisz do pliku liczby
16     int i = 1;
17     float pi = 3.1415927;
18     fprintf(fp, "liczba calkowita: %d, liczba zmiennoprzecinkowa: %f\n", i, pi);
19     fprintf(fp, "liczba calkowita: 22, liczba zmiennoprzecinkowa: 1.5\n");
20
21     //zapisz do pliku znak
22     char c = 'A';
23     fprintf(fp, "znak: %c\n", c);
24
25     //zapisz kolejne wartosci przy uzyciu petli
26     for(int i=0; i<=10; ++i)
27         fprintf(fp, "%d, %d\n", i, i*i);
28
29     //zapisz lancuch znakow
30     char tekst[] = "Hello world!";
31     fprintf(fp, "%s", tekst);
32
33     fclose(fp);
34     printf("zapisano do pliku test.txt\n");
35     return 0;
36 }
```

Przykładowy kod odczytu z pliku (kod 2 w materiałach na stronie):

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char c;
7      FILE *fp;
8      fp = fopen("test.txt", "r");
9      if(fp != NULL)
10     {
11         while((c = fgetc(fp)) != EOF)
```

```
12              printf("%c", c);
13          fclose(fp);
14      }
15      else
16      {
17          printf("brak podanego pliku!\n");
18          return 0;
19      }
20      return 0;
21  }
```

2 Funkcje zapisu do pliku

Zapis całej linijki (przy użyciu funkcji fputs) lub zapis pojedynczego znaku (przy użyciu funkcji fputc), przetestuj poniższy kod (kod 3 dostępny w materiałach na stronie):

```
1  #include <stdio.h>
2  #include<stdlib.h>
3
4  int main()
5  {
6      FILE *fp1 = fopen("plik1.txt", "w"); //identyfikator pliku
7      if(fp1 == NULL)
8      {
9          printf("blad otwierania pliku!\n");
10         return 0;
11     }
12     else
13     {
14         fputs("styczen 2023", fp1); //tekst, uchwyt do pliku 2
15         fclose(fp1);
16         printf("zapisano do plik1.txt\n");
17     }
18
19
20     FILE *fp2 = fopen("plik2.txt", "w"); //identyfikator pliku
21     if(fp2 == NULL)
22     {
23         printf("blad otwierania pliku!\n");
24         return 0;
25     }
26     else
27     {
28         fputs("s", fp2); //znak, uchwyt do pliku 2
29         fputs("\n", fp2);
30         fputs("!", fp2);
31         fclose(fp2);
32         printf("zapisano do plik2.txt\n");
33     }
34
35     return 0;
36 }
```

Poza funkcją fputs i fputc do zapisu wykorzystuje się także funkcję fprintf i służy ona do formatowanego zapisu danych. Jak nazwa wskazuje funckja ta pozwala określić typ zapisywanych danych. Przetestuj poniższy kod (kod 4 dostępny w materiałach na stronie):

```
1  #include <stdio.h>
2  #include<stdlib.h>
```

```c

int main()
{
    FILE *fp = fopen("plik3.txt", "w"); //identyfikator pliku
    if(fp == NULL)
    {
        printf("blad otwierania pliku!\n");
        return 0;
    }
    else
    {
        char znaki[] = "styczen";
        int rok = 2023;

        fprintf(fp, "%s ", znaki); //uchwyt do pliku, reszta analogiczna jak dla printf
        fprintf(fp, "%d\n", rok);

        fclose(fp);
        printf("zapisano do plik3.txt\n");
    }

    return 0;
}
```

3 Funkcje odczytu z pliku

Podobnie jak w przypadku funkcji zapisujących dane, wyróżnić można kilka funkcji odczytujących. Poniżej przykład funkcji odczytującej całą linijkę z pliku (kod 5 dostępny w materiałach na stronie). Zwróć uwagę że odczytywany jest plik, który został utworzony w ramach wcześniejszego kodu.

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

int main()
{
    char bufor[30];
    FILE *fp;

    fp = fopen("plik1.txt", "r");
    if(fp == NULL)
    {
        printf("blad otwierania pliku!\n");
        return 0;
    }
    else
    {
        //fgets(tablica do ktorej zapisze linijke, max znakow, uchwyt do pliku)
        fgets(bufor, 30, fp);
        printf("wynik odczytu z pliku: %s\n", bufor);
        fclose(fp);
    }
    return 0;
}
```

Zwróć uwagę jaka jest różnica między wywołaniem funkcji fgets a fgetc. Pierwsza w nich wczytuje całą linijkę, a druga tylko pojedyncze znaki, przetestuj poniższy kod (kod 6 dostępny w materiałach na stronie).

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

```
3
4  int main()
5  {
6      char bufor[30];
7      FILE *fp;
8
9      fp = fopen("plik1.txt", "r");
10     if(fp == NULL)
11     {
12         printf("blad otwierania pliku!\n");
13         return 0;
14     }
15     else
16     {
17         bufor[0] = fgetc(fp); //uchwyt do pliku
18         bufor[1] = fgetc(fp);
19         bufor[2] = fgetc(fp);
20
21         for(int i = 0; i < 3; i++)
22         {
23             printf("%c", bufor[i]);
24         }
25         fclose(fp);
26     }
27     return 0;
28 }
```

Odczyt z pliku może być także formatowany, przetestuj poniższy kod (kod 7 dostępny w materiałach na stronie) i zwróć uwagę, że część danych została wczytana do tablicy znaków, a część pod zmienną typu int.

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  int main()
5  {
6      char znaki[30];
7      int rok;
8      FILE *fp;
9
10     fp = fopen("plik1.txt", "r");
11     if(fp == NULL)
12     {
13         printf("blad otwierania pliku!\n");
14         return 0;
15     }
16     else
17     {
18
19         fscanf(fp, "%s %d", znaki, &rok);//uchwyt do pliku, reszta analogicznie do scanf
20
21         printf("znaki: %s, int: %d\n", znaki, rok);
22         printf("za rok bedzie: %d\n", ++rok);
23     }
24     return 0;
25 }
```

4 Zadania

Zad 1. Wykorzystaj przykładowy kod 2 z niniejszego protokołu i przerób go tak, aby program pytał użytkownika o nazwę pliku, który chce odczytać. Następnie program otwiera ten plik i wyświetla całą zawartość.

Zad 2. Utwórz plik o formacie .txt, który zawiera ciąg 10 liczb całkowitych oddzielonych spacjami. Następnie napisz program, który zapisze odczytane liczby do tablicy przy użyciu funkcji fscanf i na końcu wyświetli zawartość tej tablicy.

Zad 3. Wykorzystaj plik .txt z poprzedniego zadania i napisz program, który odczytując kolejne liczby całkowite sprawdza czy są one parzyste. Jeśli liczba jest parzysta to zapisuje ją do pliku wynikowego wykorzystując zapis formatowany, resztę liczb pomija.

Zad 4. Wykorzystaj poniższy program i zmodyfikuj go tak, aby zapisać dane osobowe dla 2 osób (utwórz tablicę struktur o rozmiarze 2), a następnie zapisz dane zawarte w tablicy struktur do pliku wynikowego. Niech plik wynikowy będzie strukturą przypominał plik o formacie .csv, a więc w pierwszej linijce zawiera informacje o nazwie kolumn oddzielone średnikami, przykładowo: "id;imie;wiek;". Kolejne wiersze tego pliku zawierają dane podane przez użytkownika, również oddzielone średnikami, przykładowo: "0;jakieś_imię;wiek_podanej_osoby;". Zarówno nagłówek jak i dane kolejnych osób powinny znaleźć się w oddzielnych wierszach.

```
1
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  struct dane_osobowe{
6        char imie[10];
7        int wiek;
8
9  }osoba;
10
11 int main()
12 {
13       printf("\nimie: ");
14       scanf("%s",osoba.imie);
15       printf("\nwiek: ");
16       scanf("%d",&osoba.wiek);
17
18       return 0;
19 }
```