70

ISO/ANSI C – dostęp do plików

- Plik to uporządkowany ciąg danych
- Dostęp do danych w pliku jest sekwencyjny, tj. istnieje pojęcie elementu aktualnego (tzw. wskaźnika pliku). Możliwy dostęp do danych w pliku jest tylko tam, gdzie wskazuje wskaźnik. Można dokonać przesunięcia wskaźnika elementu aktualnego lub też przenieść go od razu na początek lub koniec całego pliku.

- *Do elementów pliku nie można się odwoływać tak swobodnie, jak do elementów tablicy – tj. po indeksie*

71

ISO/ANSI C – dostęp do plików

FILE* - wskaźnik na strukturę zawierającą informacje o pliku, do których samodzielnie programista nigdy się nie odwołuje; FILE to *alias*, który zastępuje nazwę tej struktury.

Nigdy nie deklaruje się w programie zmiennej typu FILE. Taka zmienna tworzona jest na żądanie jako dynamiczna, a program korzysta wyłącznie ze wskaźnika na tę zmienną.

Otwieranie pliku:

```
FILE *fopen(const char *nazwapliku, const char *tryb);
```

Jeżeli zwrócona wartość nie jest NULL, to znaczy, że plik udało się pomyślnie otworzyć

72

ISO/ANSI C – dostęp do plików

FILE *fopen(const char *nazwapliku, const char *tryb);
<stdio.h>

tryby otwarcia:

r – do odczytu
w – do zapisu (jeżeli plik o podanej nazwie już istniał, to jest kasowany)
a – do pisania na końcu pliku (dołączania), jeżeli coś w nim już było zapisane (jeżeli taki plik nie istnieje – tworzy nowy)
r+ - do odczytu i zapisu (plik musi już istnieć)
w+ - do odczytu i zapisu (plik nie może jeszcze istnieć, jeżeli istnieje, to istniejąca wersja jest kasowana)
a+ - do odczytu i dołączania (jeżeli plik nie istnieje, jest tworzony)

Jaka jest różnica między 'a' i 'a+'?

73

ISO/ANSI C – dostęp do plików

FILE *fopen(const char *nazwapliku, const char *tryb);
<stdio.h>

tryby otwarcia:

r – do odczytu
w – do zapisu (jeżeli plik o podanej nazwie już istniał, to jest kasowany)
a – do pisania na końcu pliku (dołączania), jeżeli coś w nim już było zapisane (jeżeli taki plik nie istnieje – tworzy nowy)
r+ - do odczytu i zapisu (plik musi już istnieć)
w+ - do odczytu i zapisu (plik nie może jeszcze istnieć, jeżeli istnieje, to istniejąca wersja jest kasowana)
a+ - do odczytu i dołączania (jeżeli plik nie istnieje, jest tworzony)

Jaka jest różnica między 'a' i 'a+'?

Aby odpowiedzieć, najpierw należy coś wyjaśnić.

74

ISO/ANSI C – dostęp do plików

Przykład: IBM System/360. Nie ma jeszcze monitora, dlatego ..

75

ISO/ANSI C – dostęp do plików

Tryb dostępu do plików: tekstowy vs. binarny

W trybie tekstowym istnieje specjalny znak oznaczający koniec pliku (EOF):

1. Unix – Ctrl-D (kod ASCII 04)
2. Win – Ctrl-Z (kod ASCII 26)

(*ale getchar() zwróci -1, kiedy napotka EOF*)

oraz dwa znaki: CR i LF (kody ASCII: 13 i 10) razem oznaczające koniec linii.

Otwarcie pliku w trybie tekstowym sprawia, że:

1. najpierw w pliku znajdowane jest pierwsze wystąpienie EOF i usuwane. Po zakończeniu pracy w chwili zamknięcia pliku EOF jest dodawany na końcu pliku
2. każdorazowe wczytanie z pliku pary znaków CR-LF powoduje zwrócenie programowi tylko LF. Odwrotnie, wysyłanie do pliku tylko LF (znak '\n') powoduje w rzeczywistości zapisanie w pliku CR-LF

ISO/ANSI C – dostęp do plików

Tryb dostępu do plików: tekstowy vs. binarny

Otwarcie pliku w trybie binarnym sprawia, że *żadne* manipulacje ze znakami EOF ani CR-LF opisane na poprzednim slajdzie nie są wykonywane.

Uwagi:

Otwarcie w trybie binarnym może zostać wymuszone poprzez dopisanie znaku 'b' w definicji trybu otwarcia, np. 'rb', natomiast w trybie tekstowym – znaku 't', np. 'rt'

Jeżeli tryb otwarcia nie jest podany jawnie, domyślnie tryb jest odczytywany ze zmiennej globalnej '_fmode'. Domyślnym ustawieniem tej zmiennej w systemie jest wartość **_O_TEXT** ,
co oznacza tryb tekstowy.

ISO/ANSI C – dostęp do plików

Wróćmy do pytania: jaka jest różnica między 'a' i 'a+'?

'a+' pozwala również na odczyt, ale przy każdej próbie zapisu wskaźnik pliku jest automatycznie przesuwany na koniec pliku, tak aby żadne dane nie zostały utracone (*w innych trybach przejście z odczytu na zapis często wymaga ręcznego przesunięcia wskaźnika pliku – będzie o tym na następnych slajdach*).

'a' – otwiera plik w trybie pisania na końcu pliku, ale nie usuwa znaku EOF, tylko dopisuje nowe dane od razu (*jak w trybie binarnym*)
'a+' – otwiera plik w trybie czytania i dołączania, usuwa znak EOF na końcu pliku, a następnie dopisuje (jak w trybie tekstowym).
Opis wg MSDN (niejasny i częściowo nieprawdziwy)

UWAGA

W dalszej części będziemy zajmować się wyłącznie trybem tekstowym, który nie wymaga żadnych szczególnych ustawień w wywołaniach funkcji bibliotecznych bo jest trybem domyślnym.

ISO/ANSI C – dostęp do plików

Otwieranie pliku – przykład:

```
FILE* stream;
if((stream  = fopen( "crt_fopen.c", "r" )) == NULL)
      printf("Pliku 'crt_fopen.c' nie otwarto\n");
   else
      printf("Plik 'crt_fopen.c' został otwarty\n");
```


ISO/ANSI C – dostęp do plików

Otwieranie i zamykanie pliku – przykład:

```
int fclose( FILE *stream );          <stdio.h>

   FILE* stream;
   if((stream = fopen("foo.c", "r")) == NULL)
     printf("Pliku 'foo.c' nie otwarto\n" );
   else
     printf("Plik 'foo.c' został otwarty\n");

   if(fclose( stream ))
     printf("Plik 'foo.c' nie został zamknięty\n");
```

ISO/ANSI C – dostęp do plików

Formatowane czytanie z pliku otwartego w trybie tekstowym

- Po otwarciu pliku otrzymujemy prawo dostępu do tekstu w pliku.
- Dane reprezentowane są jako strumień znaków.
- Po otwarciu mamy prawo do odczytu pierwszego elementu w pliku (pierwszego znaku).
- Aktualną pozycję w pliku, z której w danej chwili mamy prawo czytać, określa *wskaźnik pliku.*
- Po odczycie wskaźnik pliku przesuwa się w przód o tyle bajtów, ile zostało odczytanych.
- Wskaźnika **nie można cofać (!)**.

ISO/ANSI C – dostęp do plików

Formatowane czytanie z pliku

```
int fscanf( FILE *stream,
    const char *format [, argument ]... );
```

```
FILE *stream;
char s[50];
if((stream = fopen( "crt_test.txt","r")) == NULL)
      exit( 0 );
fscanf( stream, "%s", s ); /* łańcuch tekstowy do
                              pierwszej spacji */
fclose( stream );
```


ISO/ANSI C – dostęp do plików

Formatowane zapisywanie do pliku

```
int fprintf( FILE *stream,
    const char *format [, argument ]    ... );
```

```
FILE *stream;
char   s1[] = "I am Groot!";
char   s2[] = "I thought it was a good plan.";
char   c = '\n';
if((stream = fopen("crt_test.txt","a+" )) == NULL)
      exit( 0 );
fprintf( stream, "%s%c", s1, c );
fprintf( stream, "%s", s2 );
fclose( stream );
```


ISO/ANSI C – dostęp do plików

Odczyt z pliku o nieznanym rozmiarze

```
int feof( FILE *stream);
```

Przykład:

```
FILE *stream;
char   buf[20];
if((stream = fopen("abc.txt", "r")) == NULL)
      exit( 0 );
fscanf(stream, "%s", buf);
while (!feof(stream)) {
  printf("%s", buf);
  fscanf(stream, "%s", buf);
}
fclose(stream);
return 0;
```


ISO/ANSI C – dostęp do plików

Odczyt z pliku o nieznanym rozmiarze

```
int feof( FILE *stream);
```

- **feof** sprawdza stan flagi w strumieniu ***stream***.
- Aby zwrócił wartość 1, flaga musi być zapalona.
- Aby flaga była zapalona, znak EOF musi zostać pobrany z pliku.

Ostatnie znaki w pliku – dwa przypadki:

1. | I | | a | m | | G | r | o | o | t | ! | CR | LF | EOF |
2. | I | | a | m | | G | r | o | o | t | ! | EOF |

```
fscanf(stream, "%s", buf); // odczyt słowa: Groot!
```

1. Pobranie znaku końca linii kończy wczytywanie słowa,
2. Pobranie znaku EOF kończy wczytywanie słowa – flaga jest zapalana (!)

ISO/ANSI C – dostęp do plików

Odczyt z pliku o nieznanym rozmiarze

```
FILE *stream;
char   buf[20];
if((stream = fopen("abc.txt", "r")) == NULL)
      exit( 0 );
fscanf(stream, "%s", buf);        // fscanf #1
while (!feof(stream)) {
  printf("%s\n", buf);
  fscanf(stream, "%s", buf);      // fscanf #2
}
fclose(stream);
return 0;
```

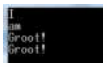
1. 

2. 

ISO/ANSI C – dostęp do plików

Odczyt z pliku o nieznanym rozmiarze

```
FILE *stream;
char   buf[20];
if((stream = fopen("abc.txt", "r")) == NULL)
      exit( 0 );
while (!feof(stream)) {
   fscanf(stream, "%s", buf);      // fscanf #1
   printf("%s\n", buf);
}
fclose(stream);
return 0;
```

1. 

2. 

88

ISO/ANSI C – dostęp do plików

Odczyt z pliku o nieznanym rozmiarze

```
FILE *stream;
char   buf[20];
if((stream = fopen("abc.txt", "r")) == NULL)
      exit( 0 );
while (!feof(stream)) {
   fscanf(stream, "%s", buf);      // fscanf #1
   printf("%s\n", buf);
}
fclose(stream);
return 0;
```

Odczyt z pustego pliku: 

Co to jest?!

89

UWAGA!

Tyle informacji wystarczy, żeby wykonać pierwsze zadania dotyczące dostępu do plików, jakie będą realizowane na laboratoriach.

Następne slajdy dotyczą dostępu do plików w sposób, jaki *może* okazać się przydatny przy realizacji zadania semestralnego.

90

ISO/ANSI C

Dostęp do plików w trybie tekstowym znak po znaku

91

ISO/ANSI C – dostęp do plików

Czytanie z pliku

```
int fgetc( FILE *stream );
```

```
<stdio.h>
```

```
FILE *stream;
   char buffer[81];
   int  i, ch;

if((stream =
  fopen("test.txt","r"))
  == NULL )
     exit( 0 );
```

```
ch = fgetc( stream );
for(i=0; (i < 80 ) &&
   (feof(stream)==0);i++)
{
      buffer[i] = (char)ch;
      ch = fgetc( stream );
}

buffer[i] = '\0';

printf( "%s\n", buffer );
fclose( stream );
```


92

ISO/ANSI C – dostęp do plików

Czytanie z pliku

```
int fgetc( FILE *stream );
                  /* zaimplementowane jako funkcja */

int getc( FILE *stream );
                  /* zaimplementowane jako funkcja i makro */
```

Działa tak samo.

93

ISO/ANSI C – dostęp do plików

Pisanie do pliku

```
int fputc( int c, FILE *stream );
```

```
FILE *stream;
char strptr[] = "Volume 2 is fine.\n";
char *p;

if( (stream = fopen( "crt_test.txt", "w" )) == NULL )
      exit( 0 );
p = strptr;

while( (*p != '\0') && fputc( *(p++), stream ) != EOF );

fclose( stream );
```


ISO/ANSI C – dostęp do plików

fgetc, getc *i* ***putc*** *są niepraktyczne, jeżeli mamy działać na danych tekstowych, reprezentujących wartości o znaczeniu leksykalnym (np. słowa, liczby).*

fgetc, getc *i* ***putc*** *wymagają w programie dodatkowego kodu służącego do analizy wczytanych znaków, aby prawidłowo określić, czy jest to np. liczba oraz czy ta liczba jest poprawnie zapisana. tymczasem są inne funkcje biblioteczne, które taką analizę mają już wbudowaną w sobie i służą np. do odczytywania całych liczb lub całych słów.*

Dlatego, o ile to możliwe, należy jednak stosować formatowany dostęp do plików.

ISO/ANSI C – dostęp do plików

Pisanie do pliku serii znaków

```
int fputs( const char *string, FILE *stream );
```

```
FILE *stream;
char strptr[] = "I am Groot!\n";

if((stream = fopen("crt_test.txt", "w")) == NULL)
      exit( 0 );

fputs( strptr, stream );

fclose( stream );
```


ISO/ANSI C – dostęp do plików

Dygresja:

```
fputs vs. puts
```

`fputs` – pisze do wskazanego strumienia (pierwszy argument wywołania)
`puts` – pisze do standardowego strumienia wyjściowego (okno konsoli)

`puts` dołącza na końcu znak nowej linii '\n'

ISO/ANSI C – dostęp do plików

Manipulowanie wskaźnikiem bieżącego elementu w pliku

ISO/ANSI C – dostęp do plików

Czytanie z pliku

- Po otwarciu pliku otrzymujemy prawo dostępu do danych w pliku
- Dane reprezentowane są jako strumień
- Po otwarciu mamy prawo do odczytu pierwszego elementu w pliku (pierwszego bajtu)
- Aktualną pozycję w pliku, z której w danej chwili mamy prawo czytać, określa wskaźnik pliku
- Po odczycie wskaźnik pliku przesuwa się w przód o tyle bajtów, ile zostało odczytanych
- Wskaźnika nie można cofać (*w zasadzie..*)

ISO/ANSI C – dostęp do plików

```
int fseek( FILE *stream, long offset, int origin );
                                                <stdio.h>
```

stream – wskaźnik do otwartego pliku

offset – liczba bajtów od miejsca wskazywanego przez origin

origin – stała:

- **SEEK_CUR** – bieżąca pozycja wskaźnika
- **SEEK_END** – koniec pliku
- **SEEK_SET** – początek pliku

fseek zwraca zero, jeżeli przesunięcie wskaźnika pliku się powiodło, lub wartość niezerową w przeciwnym przypadku

ISO/ANSI C – dostęp do plików

```
int fseek( FILE *stream, long offset, int origin );
                                                <stdio.h>
```

fseek to funkcja ryzykowna, np.:

- kiedy plik jest otwarty w trybie do dołączania ('a' lub 'a+'), bieżącą pozycją wskaźnika w pliku jest rezultat ostatniej operacji We/Wy. Jeżeli takiej operacji jeszcze nie było (plik został tylko otwarty), wskaźnik wskazuje na pierwszy element pliku (!)
- dla plików otwartych w trybie tekstowym znaki nowej linii (CR-LF) wprowadzają funkcje fseek w błąd i mogą powodować nieoczekiwane rezultaty jej działania. Poprawne działanie jest gwarantowane tylko gdy offset jest ustawione na 0 (położenie określone jest tylko przez ostatni parametr: origin).

ISO/ANSI C – dostęp do plików

```
FILE *stream;
  char line[81];
  int  result;
  stream = fopen( "fseek.out", "w+" );
  if( stream == NULL )  printf( "Plik nie został otwarty\n" );
  else {
     fprintf( stream, "fseek przesunie tutaj: abrakadabra.\n" );
     result = fseek( stream, 23L, SEEK_SET);
     if( result )  printf( "Fseek nie powiodl sie" );
     else {
        printf( "Wskaznik jest ustawiony w polowie linii.\n" );
        fscanf( stream, "%s", line );
        printf( "%s", line );
     }
     fclose( stream );
  }                    /* co pojawi się w oknie konsoli? */
```


ISO/ANSI C – dostęp do plików

`void rewind( FILE *stream );`

przesuwa wskaźnik pliku na początek pliku

Działa tak samo jak:

`fseek( stream, 0L, SEEK_SET );`

ale **rewind** nie zwraca wartości informującej czy przesunięcie wskaźnika się powiodło

ISO/ANSI C – dostęp do plików

```
FILE *stream;
  int data1, data2;
  data1 = 1;
  data2 = -37;
  if( (stream = fopen( "crt_rewind.out", "w+" )) != NULL ) {
     fprintf( stream, "%d %d", data1, data2 );
     printf( "Wartosci zapisane to: %d i %d\n", data1, data2 );
     rewind( stream );
     fscanf( stream, "%d %d", &data2, &data1 );
     printf( "Wartosci odczytane to: %d i %d\n", data1, data2 );
     fclose( stream );
  }                    /* co pojawi się w oknie konsoli? */
```


ISO/ANSI C – dostęp do plików

`int fflush( FILE *stream );`

- Wypchnięcie z tzw. strumienia (tj. bufora pliku) znajdujących się tam danych (jeżeli pomyślne, zwraca zero, jeżeli nie – `EOF`)
- Jeżeli plik był otwarty do zapisu, dane są natychmiast przepisywane do pliku. Jeżeli do odczytu – dane z bufora są tracone, a bufor staje się pusty
- `fflush(NULL)` wymusza zapisanie do plików danych ze wszystkich strumieni otwartych w trybie do zapisu
- Strumienie są zarządzane przez system operacyjny, który sam decyduje, kiedy przepisać dane z rzeczywistego pliku (np. albo w momencie zamykania pliku, albo kiedy już bufor jest zapełniony, albo też kiedy program kończy się poprawnie jednak bez wywołania polecenia zamykającego plik)

ISO/ANSI C – dostęp do plików

Oprócz strumieni otwieranych przez użytkownika istnieją trzy strumienie standardowe otwierane przez system i dostępne w programie (biblioteka `<stdio.h>`):

- **`stdin`** – standardowe wejście, np. klawiatura
- **`stdout`** – standardowe wyjście, np. okno konsoli
- **`stderr`** – standardowy strumień dla komunikatów o błędach, np. okno konsoli

Te trzy zmienne są `const` i nie można ich zapisać innymi wartościami. Można je natomiast przekierować używając funkcji **`freopen`**

ISO/ANSI C – dostęp do plików

```
FILE *freopen( const char *path,
     const char *mode, FILE *stream );
```

Zamyka plik przypisany do aktualnie otwartego strumienia i przypisuje zmienną *stream* do pliku wskazywanego przez *path*

Typowo funkcja jest używana do przekierowywania strumieni standardowych `stdin`, `stdout`, `stderr`

Argument *mode* przyjmuje takie same wartości jak w funkcji `fopen`

Przykład:

```
stream = freopen( "freopen.out", "w", stderr );
```


ISO/ANSI C – dostęp do plików

```
int fgetpos(FILE *stream, fpos_t *pos);
int fsetpos(FILE *stream, const fpos_t *pos);
```

- Odczyt aktualnej pozycji wskaźnika pliku lub przesuniecie go w nowe miejsce. Jeżeli pomyślnie, funkcja zwraca zero.
- Pozycja wskaźnika musi być przechowywana w zmiennej typu **fpos_t.**
- Pozycja wskaźnika jest interpretowana jako numer bajtu.

ISO/ANSI C – dostęp do plików

```
FILE   *stream;
fpos_t pos;
char   buffer[20];
if( (stream = fopen( "crt_fgetpos.txt", "rb" )) == NULL )
   printf( "Trouble opening file\n" );
else {
   pos = 14;
   if( fsetpos( stream, &pos ) != 0 )
        perror( "fsetpos error" );

   fread( buffer, sizeof( char ), 10, stream );
   printf( "10 bytes at byte %I64d: %.10s\n", pos, buffer );
   fclose( stream );
}
```

`sizeof()` – zwraca liczbę bajtów zajmowanych przez zmienne danego typu

ISO/ANSI C – dostęp do plików

Uwaga:

jeżeli zdecydowano otworzyć plik w trybie ‘r+’, ‘w+’ lub ‘a+’ i następuje zmiana z czynności zapisu do czynności odczytu, lub odwrotnie, to musi być najpierw wywołana funkcja **`fflush`**, **`fsetpos`**, **`fseek`**, lub **`rewind`**, która prawidłowo ustawi wskaźnik pliku.