Techniki Informatyczne

Stanisław Flaga
stanislaw.flaga@agh.edu.pl

Agenda

- Parametry wywołania programu (parametry wiersza poleceń)
- Operacje na plikach

Parametry wywołania

int main(int argc, char **argv)

int argc – liczba parametrów wywołania,

char **argv – tablica łańcuchów z parametrami
(inny zapis **char *argv[]**)

Parametry wywołania

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out To parametry \
> wywolania programu ZZ EE sd SAA 01


Program wywyolany z parametrami w liczbie: 10

Parametr argv[ 0]: ./a.out
Parametr argv[ 1]: To
Parametr argv[ 2]: parametry
Parametr argv[ 3]: wywolania
Parametr argv[ 4]: programu
Parametr argv[ 5]: ZZ
Parametr argv[ 6]: EE
Parametr argv[ 7]: sd
Parametr argv[ 8]: SAA
Parametr argv[ 9]: 01
```

Parametry wywołania

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

int main(int argc, char * argv[]){

  int licznik;

  printf("\n\nProgram wywyolany z parametrami w liczbie: %d \n\n", argc);
  for(licznik = 0; licznik < argc; licznik++)
    printf("Parametr argv[%2d]: %s\n", licznik, argv[licznik]);

  printf("\n");
  return EXIT_SUCCESS;
}
```

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define DOPOROWN "Jeden"

int main(int argc, char * argv[]){

  int licznik, porownanie;

  if(argc < 2){
    printf("\nZbyt malo parametrow wywolania. Kod wyjscia = %d\n\n", EXIT_FAILURE);
    return EXIT_FAILURE;
  }

  printf("\n\nProgram wywyolany z parametrami w liczbie: %d \n\n", argc);

  porownanie = strncmp(argv[1], DOPOROWN, strlen(DOPOROWN));
  printf("\nPorownanie\n1: %s\n2: %s \ndaje WYNIK %d\n", argv[1], DOPOROWN, porownanie);

  if(!porownanie)
    printf("\nParametr zgodny\n");
  else
    printf("\nParametr niezgodny\n\n\n");

  return EXIT_SUCCESS;
}
```

Parametry wywołania

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out

Zbyt malo parametrow wywolania. Kod wyjscia = 1
```

Parametry wywołania

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out Jeden


Program wywyolany z parametrami w liczbie: 2


Porownanie
1: Jeden
2: Jeden
daje WYNIK 0

Parametr zgodny
```

Parametry wywołania

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out Jeden


Program wywyolany z parametrami w liczbie: 2


Porownanie
1: Jeden
2: Jeden
daje WYNIK 0

Parametr zgodny
```

Parametry wywołania

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out Ieden


Program wywyolany z parametrami w liczbie: 2


Porownanie
1: Ieden
2: Jeden
daje WYNIK -1

Parametr niezgodny
```

Parametry wywołania

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out Keden


Program wywyolany z parametrami w liczbie: 2


Porownanie
1: Keden
2: Jeden
daje WYNIK 1

Parametr niezgodny
```

Parametry wywołania

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out Teden


Program wywyolany z parametrami w liczbie: 2


Porownanie
1: Teden
2: Jeden
daje WYNIK 10

Parametr niezgodny
```

Parametry wywołania

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

int main(int argc, char *argv[]){

  char *adres;
  char Lancuch[] = "To jest lancuch";
  int i = 0;

  adres = Lancuch;
  printf("\nAdres lancucha: %ld\n", (unsigned long int)adres);

  while(i < argc){
    printf("\nAdres parametru argv[%d]: %ld - %s\n", i,
           (unsigned long int)argv[i], argv[i]);
    i++;
  }

  return EXIT_SUCCESS;
}
```

Parametry wywołania

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out 1 12 123 A

Adres lancucha: 140735775774896

Adres parametru argv[0]: 140735775777831 - ./a.out

Adres parametru argv[1]: 140735775777839 - 1

Adres parametru argv[2]: 140735775777841 - 12

Adres parametru argv[3]: 140735775777844 - 123

Adres parametru argv[4]: 140735775777848 - A
```

Parametry wywołania

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out 123 4 5 6 0987654321 0

Adres lancucha: 140724014298512

Adres parametru argv[0]: 140724014302235 - ./a.out

Adres parametru argv[1]: 140724014302243 - 123

Adres parametru argv[2]: 140724014302247 - 4

Adres parametru argv[3]: 140724014302249 - 5

Adres parametru argv[4]: 140724014302251 - 6

Adres parametru argv[5]: 140724014302253 - 0987654321

Adres parametru argv[6]: 140724014302264 - 0
```

Wskaźniki

```c
#include<stdlib.h>

int main(int argc, char *argv[]){

  char *tablWskaznikow[3];
  char Str1[] = "i6";
  char Str2[] = "aBCDE";
  char Str3[] = "fgh";

  int  i;

  tablWskaznikow[0] = Str1;
  tablWskaznikow[1] = Str2;
  tablWskaznikow[2] = Str3;
  i = 0;
  printf("\nAdres 0 tablWskaznikow[%d]: %ld - %s\n", i,
      (unsigned long int)tablWskaznikow[i], tablWskaznikow[i]);
  i = 1;
  printf("\nAdres 0 tablWskaznikow[%d]: %ld - %s\n", i,
      (unsigned long int)tablWskaznikow[i], tablWskaznikow[i]);
  i = 2;
  printf("\nAdres 0 tablWskaznikow[%d]: %ld - %s\n", i,
      (unsigned long int)tablWskaznikow[i], tablWskaznikow[i]);

  return EXIT_SUCCESS;
}
```

```
sflaga@artemis ~/Dokumenty/Lab/Moje/Pliki $ ./a.out

Adres 0 tablWskaznikow[0]: 140735300045995 - i6

Adres 0 tablWskaznikow[1]: 140735300046002 - aBCDE

Adres 0 tablWskaznikow[2]: 140735300045998 - fgh
```

Pliki

do przechowywania:

- programów (kody źródłowe, pliki wynikowe, biblioteki …),
- dokumentów,
- danych dowolnego typu,
- formularzy, multimediów, ….
- …….

Pliki

Pliki tekstowe - uwaga na różne konwencje końca linii:

MAC: \r

UNIX: \n

DOS: \r\n

Pliki binarne.

Różnica między plikami tekstowymi a binarnymi.

Pliki - sposoby obsługi dostępu

Low-level I/O: korzysta z podstawowych usług udostępnianych przez system operacyjny.

Standard high-level I/O: wykorzystuje standardowy pakiet funkcji bibliotecznych języka C i definicji zawartych w pliku *stdio.h*

Standard C uwzględnia tylko standardowy pakiet wejścia-wyjścia

Łańcuchy określające tryb dla funkcji fopen()

Łańcuch	Znaczenie
`"r"`	Otwiera plik tekstowy do odczytu.
`"w"`	Otwiera plik tekstowy do zapisu, usuwając zawartość pliku, jeśli istnieje, lub tworząc nowy plik, jeśli nie istnieje
`"a"`	Otwiera plik tekstowy do zapisu, dopisując nowe dane na końcu istniejącego pliku lub tworząc nowy plik, jeśli plik nie istnieje
`"r+"`	Otwiera plik tekstowy do uaktualnienia, czyli zarówno do odczytywania, jak i zapisywania
`"w+"`	Otwiera plik tekstowy do uaktualnienia (odczytu i zapisu), usuwając zawartość pliku, jeśli istnieje, lub tworząc nowy plik, jeśli nie istnieje
`"a+"`	Otwiera plik tekstowy do uaktualnienia (odczytu i zapisu), dopisując nowe dane na końcu istniejącego pliku lub tworząc nowy plik, jeśli plik nie istnieje; odczyt może obejmować cały plik, ale zapis może polegać tylko na dodawaniu tekstu
`"rb"`, `"wb"`, `"ab"`, `"rb+"`, `"r+b"`, `"wb+"`, `"w+b"`, `"ab+"`, `"a+b"`	Jak wyżej, ale otwiera plik w trybie binarnym zamiast tekstowego
`"wx"`, `"wbx"`, `"w+x"`, `"wb+x"`, `"w+bx"`	(C11) Jak w przypadku trybów bez `x`, z tym że jeśli plik już istnieje, zostanie (w miarę możliwości) otwarty w trybie wyłącznego dostępu

getc() i putc()

Odczytanie **c** z pliku określonego wskaźnikiem **wskDoPliku**:

znak = getc(**wskDoPliku**);

Aby zapisać **znak** w pliku określonym wskaźnikiem **wskDoPliku**

putc(**znak, wskDoPliku**);

Znak końca pliku EOF (End Of File)

Program odkrywa, że plik się skończył, dopiero po próbie odczytania znaku znajdującego się już po końcu pliku.

Unikanie problemów

- Aby uniknąć problemów przy próbach odczytania pustego pliku, powinno się używać pętli **o warunku sprawdzanym na wejściu**

- **getc()** – należy podjąć próbę odczytu pierwszego znaku, **zanim program wejdzie do ciała pętli**.

- **Przykład:**

1 przyk

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

int main(void){
    int ch;          // zmienna typu int do przechowania odczytu
    FILE * fp;

    fp = fopen("a.txt", "r");
    if(fp == NULL){
        printf("\nNie mozna otworzyc pliku\n");
        return EXIT_FAILURE;
    }
    //To co poniżej powinno się dziać tylko dla przypadku fp != NULL
    ch = getc(fp);  // pobranie pierwszego znaku z wejscia

    while(ch!=EOF){
        putchar(ch);
        ch = getc(fp); // pobranie nastepnego znaku
    }
    return EXIT_SUCCESS;
}
```

2 przy

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

int main(void){
    int ch;          // zmienna typu int do przechowania odczytu
    FILE * fp;

    if ((fp = fopen("a.txt", "r")) == NULL){  //Otwarcie i sprawdzenie
                                              //poprawności otwarcia pliku
        printf("\nNie mozna otworzyc pliku\n\n");
        exit(EXIT_FAILURE);                   //wyjście bezwzględne
    }
    //To co poniżej powinno się dziać tylko dla przypadku fp != NULL

    while((ch=getc(fp)) != EOF){ // pobranie nastepnego znaku w warunku
        putchar(ch);
    }
    return EXIT_SUCCESS;
}
```

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

int main(int argc, char *argv[]){
  int znakOdczytany; // przechowuje kolejne odczytywane znaki
  unsigned long licznik = 0;

  FILE *wp;                                   // "wskaznik plikowy"

  if (argc != 2){
    printf("\nSposob uzycia: %s nazwa_pliku\n\n", argv[0]);
    return EXIT_FAILURE;
  }
  if ((wp = fopen(argv[1], "r")) == NULL){  //Otwarcie i sprawdzenie
                                            //poprawności otwarcia pliku
    printf("\nNie mozna otworzyc %s\n\n", argv[1]);
    exit(EXIT_FAILURE);                     //wyjście bezwzględne
  }
  while ((znakOdczytany = getc(wp)) != EOF){
    putc(znakOdczytany,stdout);
    licznik++;
  }
  fclose(wp);                               //Zamknięcie pliku
  printf("\nPlik %s zawiera %lu znakow\n\n", argv[1], licznik);
  return EXIT_SUCCESS;
```

fclose() – sprawdzenie poprawności zamknięcia pliku

```
if (fclose(wp) != 0)
  printf("Blad przy zamykaniu pliku %s\n", argv[1]);
```

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

int main(void){
    int ch;          // zmienna typu int do przechowania odczytu
    FILE * fp, *fzapis;

    if ((fp = fopen("a.txt", "r")) == NULL){  //Otwarcie i sprawdzenie
                                              //poprawności otwarcia pliku
        printf("\nNie mozna otworzyc pliku\n\n");
        exit(EXIT_FAILURE);                   //wyjście bezwzględne
    }

    fzapis = fopen("akopia.txt", "w");
    //To co poniżej powinno się dziać tylko dla przypadku fp != NULL

    while((ch=getc(fp)) != EOF){ // pobranie nastepnego znaku w warunku
        putc(ch, fzapis);
        putc('#', fzapis);
    }
    putc('\n', fzapis);

    fclose(fp);
    fclose(fzapis);
    return EXIT_SUCCESS;
}
```