Politechnika Białostocka
Wydział Elektryczny
Katedra Elektrotechniki Teoretycznej i Metrologii

Instrukcja
do pracowni specjalistycznej z przedmiotu

Informatyka 2

Kod przedmiotu: **ES1D300 017**

(studia stacjonarne)

JĘZYK C - ZAAWANSOWANE OPERACJE WEJŚCIA-WYJŚCIA

Numer ćwiczenia

INF28

Autor:
dr inż. Jarosław Forenc

Białystok 2017

Spis treści

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

1. Opis stanowiska

1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows (XP/ 7/10).

1.2. Oprogramowanie

Na komputerach zainstalowane jest środowisko programistyczne Microsoft Visual Studio 2008 Standard Edition lub Microsoft Visual Studio 2008 Express Edition zawierające kompilator Microsoft Visual C++ 2008.

2. Wiadomości teoretyczne

2.1. Strumienie

Operacje wejścia-wyjścia nie są elementami języka C. Zostały zrealizowane jako funkcje zewnętrzne, znajdujące się w odpowiednich bibliotekach dostarczanych wraz z kompilatorem. Funkcje te pozwalają na wykonywanie operacji wejścia-wyjścia w różny sposób i na różnym poziomie. Najczęściej do tego celu wykorzystuje się **strumienie**.

Strumień (ang. *stream*) jest pojęciem abstrakcyjnym. Jego nazwa bierze się z analogii między przepływem danych, a np. wody. W strumieniu dane płyną od źródła do odbiorcy. Zadaniem użytkownika jest określenie źródła i odbiorcy, wybranie typu danych oraz sposobu ich przesyłania. Strumień może być skojarzony ze zbiorem danych na dysku (np. plik) lub zbiorem danych pochodzących z urządzenia znakowego (np. klawiatura). Niezależnie od fizycznego medium, z którym strumień jest skojarzony, wszystkie strumienie mają podobne właściwości.

W języku C strumienie reprezentowane są przez zmienne będące wskaźnikami do struktur typu **FILE**. Definicja struktury **FILE** znajduje się w pliku nagłówkowym **stdio.h**:


```
struct _iobuf {
        char *_ptr;
        int   _cnt;
        char *_base;
        int   _flag;
        int   _file;
        int   _charbuf;
        int   _bufsiz;
        char *_tmpfname;
        };
typedef struct _iobuf FILE;
```

Podczas pisania programów nie ma potrzeby bezpośredniego odwoływania się do pól powyższej struktury.

Gdy program w języku C rozpoczyna działanie automatycznie tworzone są i otwierane trzy standardowe strumienie wejścia-wyjścia:

- **stdin** - standardowe wejście, skojarzone z klawiaturą;
- **stdout** - standardowe wyjście, skojarzone z ekranem monitora;
- **stderr** - standardowe wyjście dla komunikatów o błędach, domyślnie skojarzone z ekranem monitora.

Definicje standardowych strumieni umieszczone są w pliku nagłówkowym **stdio.h**:

```
_CRTIMP FILE * __cdecl __iob_func(void);

#define stdin  (&__iob_func()[0])
#define stdout (&__iob_func()[1])
#define stderr (&__iob_func()[2])
```

W dotychczas pisanych programach korzystano już z tych strumieni. Podczas każdego wywołania funkcji **printf()** niejawnie wykorzystywany był strumień **stdout**, a podczas wywołania funkcji **scanf()** - strumień **stdin**.

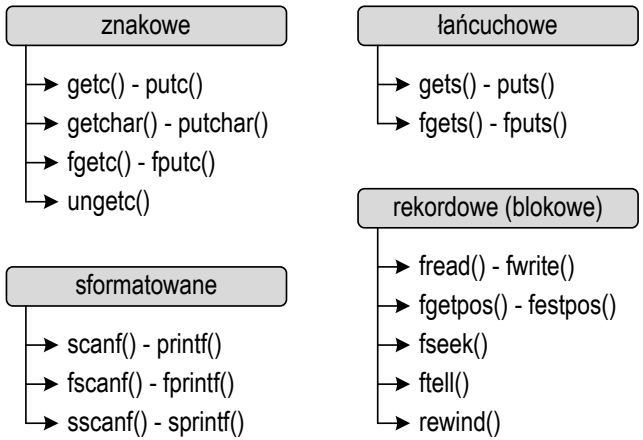
Tworzenie własnych strumieni oraz wykonywanie operacji na plikach opisane zostało w instrukcjach **INF29** i **INF30** do przedmiotu Informatyka 2.

2.2. Typy operacji wejścia-wyjścia

Operacje wejścia-wyjścia można podzielić na cztery typy:

- **znakowe** - przetwarzanie danych odbywa się znak po znaku;
- **łańcuchowe** - przetwarzanie danych odbywa się wierszami;
- **sformatowane** - przy przetwarzaniu danych stosowane są specyfikatory formatu;
- **rekordowe (blokowe)** - dane przetwarzane są całymi blokami (rekordami).

Nazwy wybranych funkcji wykonujących poszczególne typy operacji przedstawiono na Rys. 1. Zastosowanie tych funkcji w programie wymaga dołączenia pliku nagłówkowego **stdio.h**.



Rys. 1. Typy operacji wejścia-wyjścia w języku C

W kolejnych rozdziałach przedstawiono opis wybranych funkcji wykonujących operacje znakowe, łańcuchowe i sformatowane.

2.3. Operacje znakowe

`getc()`	Nagłówek: int **getc (FILE *stream);**

- funkcja **getc()** pobiera jeden znak z aktualnej pozycji otwartego strumienia **stream** i uaktualnia pozycję;
- zmienna **stream** powinna wskazywać strukturę **FILE** reprezentującą strumień skojarzony z otwartym plikiem lub jeden ze standardowo otwartych strumieni (np. **stdin**);
- funkcja zwraca wartość całkowitą kodu wczytanego znaku lub wartość **EOF**, jeśli wystąpił błąd lub przeczytany został znacznik końca pliku;
- przykład odczytania jednego znaku ze standardowego wejścia (klawiatury) i jego wyświetlenia na ekranie:

```
int znak;
znak = getc(stdin);
printf("%c",znak);
```

`putc()`	Nagłówek: int **putc(**int **znak, FILE *stream);**

- funkcja **putc()** wpisuje znak do otwartego strumienia reprezentowanego przez argument **stream**;
- zmienna **stream** powinna wskazywać strukturę **FILE** reprezentującą strumień skojarzony z otwartym plikiem lub jeden ze standardowo otwartych strumieni (np. **stdout**);
- jeśli wykonanie zakończyło się poprawnie, to zwraca wypisany **znak**; jeśli wystąpił błąd, to zwraca wartość **EOF**;
- przykład wyświetlenia jednego znaku na standardowym wyjściu (ekranie monitora):

```
int znak = 'x';
putc(znak,stdout);
```

`getchar()`	Nagłówek: int **getchar**(void);

- funkcja **getchar()** pobiera znak ze strumienia **stdin** (klawiatura);
- jeśli wykonanie zakończyło się poprawnie, to zwraca przeczytany znak;
- jeśli wystąpił błąd albo został przeczytany znacznik końca pliku, to zwraca wartość **EOF**;
- funkcja **getchar()** jest równoważna wywołaniu funkcji **getc(stdin)**;
- przykład odczytania jednego znaku ze standardowego wejścia (klawiatury) i jego wyświetlenia na ekranie:

```
int znak;
znak = getchar();
printf("%c",znak);
```

`putchar()`	Nagłówek: int **putchar**(int **znak**);

- funkcja **putchar()** wpisuje **znak** do strumienia **stdout** (standardowo ekran);
- jeśli wykonanie zakończyło się poprawnie, to zwraca wypisany **znak**;
- jeśli wystąpił błąd, to zwraca wartość **EOF**;
- funkcja **putchar()** jest równoważna funkcji **putc()** wywołanej z drugim argumentem równym **stdout**;
- przykład wyświetlenia jednego znaku na standardowym wyjściu (ekranie monitora):

```
int znak = 'x';
putchar(znak);
```


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

- funkcja **fgetc()** pobiera jeden znak ze strumienia wskazywanego przez **stream**;
- jeśli wykonanie zakończyło się poprawnie, to zwraca przeczytany znak po przekształceniu go na typ **int**;
- jeśli wystąpił błąd lub został przeczytany znacznik końca pliku, to zwraca wartość **EOF**.

`fputc()`	Nagłówek: int **fputc**(int **znak**, FILE *stream);

- funkcja **fputc()** wpisuje **znak** do otwartego strumienia reprezentowanego przez argument **stream**;
- jeśli wykonanie zakończyło się poprawnie, to zwraca wypisany **znak**;
- jeśli wystąpił błąd to zwraca wartość **EOF**.

`ungetc()`	Nagłówek: int **ungetc**(int **znak**, FILE *stream);

- funkcja **ungetc()** umieszcza **znak** z powrotem w strumieniu wejściowym.

2.4. Operacje łańcuchowe

`gets()`	Nagłówek: char* **gets**(char ***s**);

- funkcja **gets()** pobiera do bufora pamięci wskazywanego przez argument **s** linię znaków ze strumienia **stdin** (standardowo klawiatura);
- wczytywanie jest kończone po napotkaniu znacznika nowej linii **'\n'**, który w buforze pamięci **s** zastępowany jest znakiem końca łańcucha **'\0'**;
- **gets()** umożliwia wczytanie łańcucha zawierającego spacje i tabulatory;

- jeśli wykonanie zakończyło się poprawnie, to zwraca wskazanie do łańcucha **s**; jeśli napotka znacznik końca pliku lub gdy wystąpił błąd, to zwraca **EOF**;
- przykład odczytania jednego wiersza tekstu z klawiatury:

```
char tablica[80];
gets(tablica);
```

`puts()`	Nagłówek: int **puts**(const char ***s**);

- funkcja **puts()** wpisuje łańcuch **s** do strumienia **stdout** (standardowo ekran), zastępując znak **'\0'** znakiem **'\n'** (co oznacza automatyczne przejście do nowego wiersza po wyświetleniu zawartości łańcucha **s**);
- jeśli wykonanie zakończyło się poprawnie, to funkcja **puts()** zwraca ostatni wypisany znak; jeśli wystąpił błąd, to zwraca wartość **EOF**;
- przykład wyświetlenia jednego wiersza tekstu:

```
char tablica[30] = "Programowanie nie jest trudne";
puts(tablica);
```

`fgets()`	Nagłówek: char* **fgets**(char ***buf**, int **max**, FILE ***stream**);

- funkcja **fgets()** pobiera znaki z otwartego strumienia reprezentowanego przez **stream** i zapisuje je do bufora pamięci wskazanego przez **buf**;
- pobieranie znaków jest przerywane po napotkaniu znacznika końca linii **'\n'** lub odczytaniu **max-1** znaków;
- po ostatnim przeczytanym znaku wstawia do bufora **buf** znak **'\0'**;
- jeśli wykonanie zakończyło się poprawnie, to zwraca wskazanie do łańcucha **buf**; jeśli napotka znacznik końca pliku albo gdy wystąpił błąd, to zwraca wartość **NULL**.

`fputs()`	Nagłówek: int **fputs**(const char ***buf**, FILE ***stream**);

- funkcja **fputs()** wpisuje łańcuch **buf** do strumienia **stream**, nie dołącza znaku końca wiersza '**\n**';
- jeśli wykonanie zakończyło się poprawnie, to zwraca ostatni wypisany znak; jeśli wystąpił błąd, to zwraca wartość **EOF**.

2.5. Operacje sformatowane

`scanf()`	Nagłówek: int **scanf**(const char ***format, ...**);

- funkcja **scanf()** wprowadza dane ze strumienia **stdin** zgodnie z podanymi specyfikatorami formatu.

`fscanf()`	Nagłówek: int **fscanf**(FILE ***stream**, const char ***format, ...**);

- funkcja **fscanf()** działa podobnie jak **scanf()**, ale czyta dane z otwartego strumienia **stream** do listy argumentów.

`sscanf()`	Nagłówek: int **sscanf**(const char ***buf**, const char ***format, ...**);

- funkcja **sscanf()** działa podobnie jak **scanf()**, ale czyta dane z bufora pamięci **buf** do listy argumentów.

`printf()`	Nagłówek: int **printf**(const char ***format, ...**);

- funkcja **printf()** wyprowadza dane do strumienia **stdout** (standardowo ekran monitora) zgodnie z podanymi specyfikatorami formatu.

`fprintf()`	Nagłówek: int **fprintf(FILE *stream,** const char ***format, ...);**

- funkcja **fprintf()** działa podobnie jak **printf()**, ale wyprowadza dane do otwartego strumienia **stream**.

`sprintf()`	Nagłówek: int **sprintf(**char ***buf,** const char ***format, ...);**

- funkcja **sprintf()** działa podobnie jak **printf()**, ale wyprowadza dane do bufora pamięci wskazywanego przez **buf**.

3. Przebieg ćwiczenia

Na pracowni specjalistycznej należy wykonać zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane inne zadania.

1. Wczytaj z klawiatury datę w formacie **dd-mm-rrrr**. Zadeklaruj trzy zmienne (**d**, **m**, **r**) i przypisz im dzień, miesiąc i rok zawarte w dacie. Wyświetl otrzymane wartości na ekranie.

   Przykładowe wywołanie programu:

```
Podaj date: 21-09-2015
----------------------
Dzien:    21
Miesiac:  9
Rok:      2015
```

2. Napisz program, w którym użytkownik wprowadza z klawiatury cenę **netto** pewnego towaru. Zakładając stawkę podatku VAT równą 23% oblicz cenę **brutto** oraz kwotę podatku **VAT**. Wykorzystując funkcję **sprintf()** utwórz jeden łańcuch znaków zawierający cenę **netto**, kwotę **VAT** i cenę **brutto**. Zastosuj poniższy wzór wydruku (dla wprowadzonej ceny netto 100 PLN):

```
"Netto: 100.00 PLN, VAT: 23.00 PLN, Brutto: 123.00 PLN"
```

   Wyświetl otrzymany łańcuch znaków na ekranie stosując funkcje: **printf()**, **fprintf()**, **puts()**, **fputs()**, **putc()**, **fputc()**, **putchar()**.

3. Napisz program działający w pętli, w którym użytkownik wprowadza z klawiatury numery **PESEL**. Program powinien wyświetlić datę urodzenia oraz informację, czy dany numer PESEL należy do kobiety czy do mężczyzny. Wyjście z programu powinno nastąpić, gdy użytkownik nie poda numeru, a tylko wciśnie klawisz **Enter**.

   Przykład uruchomienia programu:

```
Podaj PESEL: 92040251610
PESEL 92040251610, 02-04-1992, mezczyzna
----------------------------------------
Podaj PESEL: 05310308202
PESEL 05310308202, 03-11-2005, kobieta
----------------------------------------
Podaj PESEL:
Koniec
```

4. W budynku znajduje się system pomiarowy, który w trzech punktach dokonuje pomiaru dwóch wielkości: temperatury i wilgotności powietrza. Wyniki pomiarów są przesyłane w postaci tekstowej do stacji bazowej drogą radiową. Przesyłany komunikat ma następujący format:

   $P nr typ = wartość

   gdzie **nr** jest numerem punktu pomiarowego (**1**, **2** lub **3**), **typ** określa mierzoną wielkość (**TEMP** - temperatura, **HUM** - wilgotność), zaś **wartość**, to wynik pomiaru. Przykład:

```
$P1HUM = 39.22    - pomiar wilgotności w punkcie nr 1;
$P3TEMP = 26.19   - pomiar temperatury w punkcie nr 3;
```

   Napisz funkcję **analiza()**, która jako argument otrzymuje komunikat i wyświetla na ekranie szczegółowe informacje o pomiarze. Umieść w programie poniższą deklarację i inicjalizację tablicy znaków.

```
char kom[8][20] = {"$P1HUM = 39.22"  ,"$P3TEMP = 26.19",
                   "$P2TEMP = 24.33" ,"$P3HUM = 65.14",
                   "$P1TEMP = 29.56" ,"$P1HUM = 85.91",
                   "$P2HUM = 68.23"  ,"$P3TEMP = 28.74"};
```

   Wywołaj napisaną funkcję dla kolejnych komunikatów zawartych w tablicy. Poniżej pokazano przykładowe wywołanie funkcji **analiza()** i otrzymane wyniki.

```
for (int i=0; i<8; i++)
    analiza(kom[i]);
```

```
Punkt nr 1, pomiar wilgotnosci, wynik: 39.22 [%]
Punkt nr 3, pomiar temperatury, wynik: 26.19 [C]
Punkt nr 2, pomiar temperatury, wynik: 24.33 [C]
Punkt nr 3, pomiar wilgotnosci, wynik: 65.14 [%]
Punkt nr 1, pomiar temperatury, wynik: 29.56 [C]
Punkt nr 1, pomiar wilgotnosci, wynik: 85.91 [%]
Punkt nr 2, pomiar wilgotnosci, wynik: 68.23 [%]
Punkt nr 3, pomiar temperatury, wynik: 28.74 [C]
```

5. Napisz program wczytujący ułamek zwykły i redukujący go do najmniejszego mianownika. Przykład działania programu:

```
Ulamek:       5/15
Po redukcji:  1/3
```

4. Literatura

[1] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.

[2] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.

[3] Prinz P., Crawford T.: Język C w pigułce. APN Promise, Warszawa, 2016.

[4] King K.N.: Język C. Nowoczesne programowanie. Wydanie II. Helion, Gliwice, 2011.

[5] Kochan S.G.: Język C. Kompendium wiedzy. Wydanie IV. Helion, Gliwice, 2015.

[6] http://www.cplusplus.com/reference/clibrary - C library - C++ Reference

5. Zagadnienia na zaliczenie

1. Wyjaśnij pojęcie strumienia.
2. Omów standardowe strumienie wejścia-wyjścia.
3. Scharakteryzuj typy operacji wejścia-wyjścia w języku C.

6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.
- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.
- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.
- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.
- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.

- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.