Języki programowania

Obsługa plików

Obsługa plików

- I znowu – można jak w C, za pomocą „starych” struktur i metod:

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

int main ()
{
  FILE *fp; /* używamy metody wysokopoziomowej - musimy mieć zatem identyfikator pliku, uwaga na gwiazdkę! */
  char tekst[] = "Hello world";
  if ((fp=fopen("test.txt", "w"))==NULL) {
    printf ("Nie mogę otworzyć pliku test.txt do zapisu!\n");
    exit(1);
    }
  fprintf (fp, "%s", tekst); /* zapisz nasz łańcuch w pliku */
  fclose (fp); /* zamknij plik */
  return 0;
}
```

… ale my się uczymy C++ i mamy strumienie:

Zapis (ofstream)

```
4  int main()
5  {
6     char znak;
7     std::ofstream plik("testowy.txt");
8     do {
9        znak = std::cin.get();
10       plik << znak;
11    } while (znak != '.');
12    return 0;
13 }
```

Czego tu wszędzie brakuje?

```
plik.close();
```

Odczyt (ifstream)

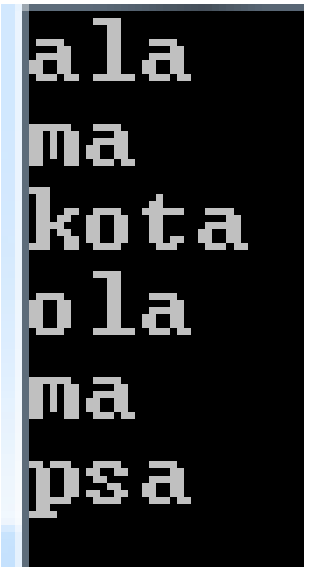
```
1  #include <iostream>
2  #include <fstream>
3
4  int main()
5  {
6     char znak;
7     std::ifstream plik("testowy.txt");
8     do {
9        znak = plik.get();
10       std::cout << znak;
11    } while (znak != '.');
12
13 system("PAUSE");
14 return 0;
15 }
```

```
6  std::string linia;
7  std::ifstream plik("testowy.txt");
8
9     do {
10       getline(plik,linia);
11       std::cout << linia <<std::endl;
12    } while (!plik.eof());
```

Odczyt pliku typowo jak strumień

```
1  #include <iostream>
2  #include <fstream>
3
4  using namespace std;
5
6  int main()
7  {
8      string linia;
9      ifstream plik;
10     plik.open("testowy.txt");
11     do {
12         plik >> linia;
13         cout << linia << endl;
14     } while (!plik.eof());
15     plik.close();
16     return 0;
17 }
```

Wynik



Plik testowy.txt

```
1  ala ma kota
2  ola ma psa
```

Pliki i strumienie

```
1  #include <iostream>
2  #include <fstream>
3
4  int main()
5  {
6  std::string linia;
7  std::ofstream plik;
8  plik.open("testowy.txt",std::ofstream::app);
9  for(int i=0;i<10;++i) {
10 plik << rand()%10 <<std::endl;
11 }
12 std::ifstream plikO;
13 plikO.open("testowy.txt");
14    do {
15       getline(plikO,linia);
16       std::cout << linia <<std::endl;
17    } while (!plikO.eof());
18
19 system("PAUSE");
20 return 0;
21 }
```

app – jak append

http://www.cplusplus.com/reference/iostream/ifstream/

http://www.cplusplus.com/reference/iostream/ofstream/

Pliki i strumienie – lepiej tak

```
#include <iostream>
#include <fstream>

int main()
{
std::string linia;
std::ofstream plik;
plik.open("testowy.txt",std::ofstream::app);
for(int i=0;i<10;++i) {
plik << rand()%10 <<std::endl;
}
std::ifstream plikO;
plikO.open("testowy.txt");
   while(plik.good()){
      getline(plikO,linia);
      std::cout << linia <<std::endl;
   }

system("PAUSE");
return 0;
}
```

Funkcja good() jest tutaj lepsza.

Odczyt

- **int get();** – zwraca znak ze strumienia (lub EOF jeśli skończył się plik) i konwertuje na typ integer.
- **istream& get(char& c);**
  – wpisuje do c odczytany ze strumienia znak [koniec pliku należy sprawdzić przez eof()].
- **istream& get(char* buf, int len, char eot = '\n');** – pobiera ze strumienia do `buf' maksymalnie `len' znaków i aż do napotkania znaku `eot' lub do pobrania len – 1 znaków.
- **istream& getline(char* buf, int len, char eot = '\n');** – jak powyższa get, ale znak `eot' nie jest zapisywany w `buf'
- **istream& read(char* ptr, long len);**
- **istream& read(void* ptr, long len);**
  – odczytują ze strumienia maksymalnie `len' znaków do `ptr';
- **istream& unget();**– wstawia znak z powrotem do strumienia
- **bool good()** – sprawdza czy nie zostały ustawione jakiekolwiek flagi błędów dla strumienia (*eofbit*, *failbit* and *badbit*).

Ćwiczenia

- Dokonaj normalizacji danych w pliku. Załóżmy, że plik input.txt wygląda tak:

1. Dla każdego przypadku testowego wyświetl średnią z trzeciej kolumny.
2. Znormalizuj wszystkie dane w trzeciej kolumnie do przedziału [0…1].

```
1  Case1   1   0.12
2  Case2   1   1.90
3  Case1   2   100.89
4  Case3   1   645.1
5  Case2   2   44.9
6  Case4   1   0.33333333
7  Case1   3   1.12
8  Case1   4   0.07
```

- Normalizacja:

$$y = y_{min} + \frac{(x - x_{min}) \cdot (y_{max} - y_{min})}{x_{max} - x_{min}}$$

Ymin = 0
Ymax = 1

Ćwiczenia – hasła

- Informatyk z firmy „KompOK” zapisał w pliku hasla.txt 200 haseł. Każde hasło umieszczone jest w osobnym wierszu pliku. Hasło składa się tylko z małych liter alfabetu angielskiego, zaś jego długość wynosi od 3 do 10 znaków.
- Wykorzystując dane zawarte w tym pliku, wykonaj poniższe polecenia. Odpowiedzi do poszczególnych podpunktów zapisz w pliku tekstowym odpowiedzi1.txt opatrując je numerem podpunktu:

Pliki z danymi: www.tomaszx.pl/materialy/jpoig_dane.zip

Ćwiczenia hasła – c.d.

1. Podaj, ile haseł ma parzystą, a ile nieparzystą liczbę znaków.
2. Utwórz zestawienie haseł (po jednym w wierszu), które są palindromami.
3. Utwórz zestawienie haseł (po jednym w wierszu) zawierających w sobie dwa kolejne znaki, których suma kodów ASCII wynosi 220.
4. Wypisz najkrótsze i najdłuższe hasło z pliku.

Przykłady:

- Hasło *krzysio* zawiera dwa kolejne znaki *si*, których suma kodów ASCII wynosi 220. Kod ASCII znaku *s* to 115, kod znaku *i* to 105; suma kodów wynosi 115+105 = 220.
- Hasło *cyrk* zawiera również takie dwa kolejne znaki. Kod ASCII znaku *c* to 99, kod ASCII znaku *y* to 121; suma kodów wynosi 99+121=220

Ćwiczenia liczby

- W pliku liczby.txt, w oddzielnych wierszach, znajduje się 1000 liczb zapisanych w systemie dwójkowym o długościach zapisów od 2 do 16 cyfr (0 lub 1).
- Napisz program, którego wykonanie da odpowiedzi do poniższych podpunktów. Odpowiedzi zapisz w pliku odpowiedzi2.txt, a każdą odpowiedź poprzedź literą oznaczającą ten podpunkt.

1. Ile jest liczb parzystych w całym pliku?
2. Jaka jest największa liczba w tym pliku? Podaj jej wartość w dwóch systemach: dwójkowym i dziesiętnym.
3. Ile liczb w całym pliku ma dokładnie 9 cyfr? Podaj sumę tych liczb w systemie dwójkowym i dziesiętnym.

Ćwiczenia – HTML

- Napisać program, którego zadaniem jest odczytanie danych tabelarycznych w pliku tekstowym, a następnie zapisanie ich do nowego pliku w postaci kodu HTML. Dane rozdzielone spacją.

Przykład:

Wejście:

"Waga" "Wzrost" "BMI" "Nadwaga"
70 1,8 21,6 "NIE"
67 1,77 21,39 "NIE"
85 1,7 29,41 "TAK"
100 1,92 27,13 "TAK"

Wynik:

```
<html><body>
<table>
<tr><td>"Waga"</td><td>"Wzrost"</td><td>"BMI"</td><td>"Nadwaga"</td>
</tr>
<tr><td>70</td><td>1,8</td><td>21,6</td><td>"NIE"</td></tr>
<tr><td>67</td><td>1,77</td><td>21,39</td><td>"NIE"</td></tr>
<tr><td>85</td><td>1,7</td><td>29,41</td><td>"TAK"</td></tr>
<tr><td>100</td><td>1,92</td><td>27,13</td><td>"TAK"</td></tr>
</table>
</body></html>
```

Ćwiczenia – pensja

Napisać funkcję **przepisz**, która jako **pierwszy** parametr otrzymuje **nazwę pliku tekstowego**, w którym każda linia wygląda następująco:

- `imię*nazwisko*plec*wiek*pensja`

gdzie imię i nazwisko zapisane są literami alfabetu angielskiego, płeć to litera 'K' lub 'M', a wiek i pensja, to liczby całkowite dodatnie. Liczba osób zapisanych w pliku jest nieokreślona.

W rezultacie jej działania powinny powstać dwa pliki wyjściowe, gdzie jeden będzie zawierał jedynie kobiety (nazwa taka sama jak pliku w przypadku wejściowego poprzedzona, literą 'k'), a drugi mężczyzn (poprzedzony literą 'm').

Jednocześnie do nowych plików należy:

- nie przepisywać oznaczenia płci,
- kobietom o wieku większym niż podany jako **drugi parametr** (wiek) podnieść pensję o 15%,
- mężczyzną podnieść pensję o tyle, ile mają lat.

W wyniku działania funkcji powinien zostać zwrócony **średni wiek wszystkich mężczyzn** z pliku wejściowego.