

```
1 // process_text_stream.cpp
2 //
3 #ifdef _WIN32
4     #pragma warning(disable:4996)
5     #include <tchar.h>
6     #include <windows.h>
7     #include <conio.h>
8     #include <direct.h>
9 #elif (defined __linux__) || (defined _AIX)
10    #include <stdlib.h>
11    #include <sys/types.h>
12    #include <sys/stat.h>
13    #include <unistd.h>
14    typedef char _TCHAR;
15    #define _tmain main
16 #endif
17
18 #include <stdio.h>
19 #include <string.h>
20 #include <fstream>
21 #include <iostream>
22 #include <iomanip>
23 using namespace std;
24
25 #include "Veta.h"
26
27 #define MAXLINE 255
28
29 void my_getch();
30 void GetNamesOfIOFiles(const char[], char[]);
31 //-----
32 int _tmain(int argc, _TCHAR* argv[])
33 {
34     ifstream in;
35     VetaClass** veta;
36     unsigned i, j, k, n, PocetViet;
37     char c, * str, nameoffile[MAXLINE];
38
39     cout << "\n Tento program nacita vetu z textoveho suboru, vyhodi z nej "
40          << "prebytocne\nmedzery a na obrazovku vypise upravenu vetu a potom"
41          << " pod seba jednotlivne slova\nspolu s ich dlzkami.\n\n";
42
43     GetNamesOfIOFiles("VETY.TXT", nameoffile);
44     in.open(nameoffile, ios::in);
45     if (!in) {
46         cout << " Subor VETY.TXT sa nepodarilo otvorit.\n";
47         my_getch();
48         exit(1);
49     }
50
51     i = 0;
52     veta = new VetaClass * [50];
53     do {
```

```
54     do {
55         in.get(c);
56     } while ((!isalnum(c)) && (!ispunct(c)) && (!in.eof()));
57
58     if ((isalnum(c) || (ispunct(c)))) {
59         veta[i] = new VetaClass;
60         in.putback(c);
61         in >> veta[i];
62         i++;
63     }
64 } while (!in.eof());
65 PocetViet = i;
66 in.close();
67
68 cout << "  Zadana veta:\n\n";
69 cout << "  ";
70 for (i = 0; i < PocetViet; i++)
71     cout << veta[i];
72 cout << "\n\n  Press any key!";
73 my_getch();
74
75 n = 0;
76 cout << "\n\n  Slova vety          Dlzka slov\n";
77 for (i = 0; i < PocetViet; i++) {
78     for (j = 0; j < veta[i]->PocetSlov; j++) {
79         str = veta[i]->DajSlovo(j);
80         k = (unsigned)strlen(str);
81         cout << "\n  " << str;
82         cout.setf(ios::right);
83         cout << setw(27 - k) << k;
84         n++;
85         if (n % 40 == 0) {
86             n = 0;
87             cout << "\n\n  Press any key!\n";
88             my_getch();
89         }
90     }
91 }
92 cout << endl;
93
94 for (i = 0; i < PocetViet; i++)
95     delete veta[i];
96 delete[] veta;
97
98 my_getch();
99 return 0;
100 }
101 //-----
102 void my_getch()
103 {
104 #ifdef _WIN32
105     _getch();
106 #else
```

```
107     cout << endl;
108 #endif
109 }
110 //-----
111 void GetNamesOfIOFiles(const char name_of_input_file[], char path_to_input_file[])
112 {
113     char current_path[MAXLINE];
114     current_path[0] = '\0';
115
116 #ifdef _WIN32
117     TCHAR exePath[MAXLINE];
118
119     HMODULE hModule = GetModuleHandle(NULL);
120     if (hModule != NULL) {
121         if (!GetModuleFileName(hModule, exePath, MAXLINE)) {
122             cout << "Nepodarila sa zistit cesta k exe-suboru.\n";
123             my_getch();
124             exit(1);
125         }
126     }
127     else {
128         cout << "Module handle is NULL.\n" << endl;
129         my_getch();
130         exit(1);
131     }
132 #ifdef __BORLANDC__
133     int iii;
134     bool flag = false;
135     for (iii = strlen(exePath); iii >= 0; iii--) {
136         if (!flag && exePath[iii] == '\\') {
137             current_path[iii + 1] = '\0';
138             flag = true;
139         }
140         if (flag)
141             current_path[iii] = (char)exePath[iii];
142     }
143 #else
144     int iii;
145     bool flag = false;
146     for (iii = (int)wcslen(exePath); iii >= 0; iii--) {
147         if (!flag && exePath[iii] == '\\') {
148             current_path[iii + 1] = '\0';
149             flag = true;
150         }
151         if (flag)
152             current_path[iii] = (char)exePath[iii];
153     }
154 #endif
155 #elif (defined __linux__)
156     unsigned iii;
157     char line[MAXLINE];
158     FILE* fp;
159     if ((fp = popen("/bin/pwd", "r")) == NULL) {
```

```
160     perror("popen error");
161     exit(1);
162 }
163 if (fgets(line, MAXLINE, fp) == NULL) {
164     perror("fgets error");
165     exit(1);
166 }
167 pclose(fp);
168
169 iii = 0;
170 while (line[iii] != '\r' && line[iii] != '\n') {
171     current_path[iii] = line[iii];
172     iii++;
173 }
174 current_path[iii] = '\0';
175 #elif (defined _AIX)
176     unsigned iii;
177     char line[MAXLINE];
178     FILE* fp;
179     if ((fp = popen("user/bin/pwd", "r")) == NULL) {
180         perror("popen error");
181         exit(1);
182     }
183     if (fgets(line, MAXLINE, fp) == NULL) {
184         perror("fgets error");
185         exit(1);
186     }
187     pclose(fp);
188
189     iii = 0;
190     while (line[iii] != '\r' && line[iii] != '\n') {
191         current_path[iii] = line[iii];
192         iii++;
193     }
194     current_path[iii] = '\0';
195 #endif
196
197     path_to_input_file[0] = '\0';
198     strcat(path_to_input_file, current_path);
199 #if (defined __linux__) || (defined _AIX)
200     strcat(path_to_input_file, "/inputs/");
201 #elif (defined _WIN32)
202     strcat(path_to_input_file, "inputs\\");
203 #endif
204     strcat(path_to_input_file, name_of_input_file);
205 }
206 //-----
207
```