1 Pierwsze kroki w C++ cz.3

2 Obsługa plików

Do pracy z plikami zewnętrznymi niezbędna będzie biblioteka **fstream**. Udostępnia ona programiście narzędzia do zapisu i odczytu plików.

```
#include <fstream>
```

2.1 Typ zmiennej fstream.

Przed rozpoczęciem pracy na pliku niezbędne jest stworzenie zmiennej, która pozwoli nam na dokonywanie operacji na wybranym pliku:

```
std::fstream plik;
```

lub w przypadku korzystania z przestrzeni nazw std:

```
fstream plik;
```

Utworzona w ten sposób zmienna nie wskazuje na żaden plik. W celu przypisania do niej konkretnego pliku należy skorzystać z funkcji **open()** zawartej w klasie **std::fstream**:

```
plik.open( "nazwa_pliku.txt", tryb_otwarcia_pliku);
```

Podczas przypisywania pliku do zmiennej należy wyszczególnić jeden, lub więcej trybów otwarcia pliku:
ios::app — Ustawia wewnętrzny wskaźnik zapisu pliku na jego koniec. Plik otwarty w trybie tylko do zapisu. Dane mogą być zapisywane tylko i wyłącznie na końcu pliku.
ios::ate — Ustawia wewnętrzny wskaźnik pliku na jego koniec w chwili otwarcia pliku.
ios::binary — Informacja dla kompilatora, aby dane były traktowane jako strumień danych binarnych.
ios::in — Zezwala na odczyt danych z pliku.
ios::out — Zezwala na zapis danych do pliku.
ios::trunc — Zawartość pliku zostaje wyczyszczona podczas otwarcia.

```
fstream plik;
plik.open( "nazwa_pliku.txt", ios::in | ios::out );
```

Po zakończeniu operacji z plikiem należy go zamknąć. W tym celu wykorzystana zostanie funkcja **close()**:

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

Przykład:

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    fstream plik;
    plik.open( "nazwa_pliku.txt", ios::in | ios::out );
    if( plik.good() == true ) {
       cout << "Uzyskano dostep do pliku!" << endl;

        plik.close();
    }
    else cout << "Brak dostepu do pliku " << endl;

    return( 0 );
}
```

2.2 Odczytywanie danych z pliku.

Istnieje kilka metod na odczytywanie danych z pliku.

2.2.1 Odczyt danych za pomocą strumienia.

Odczyt danych za pomocą strumienia jest analogiczny do działania **std::cin»**. Dane odczytane w ten sposób są zawsze traktowane jako tekst. Niestety funkcja ta nie pozwoli na odczytanie informacji o białych znakach (podobnie jak w rozdziale 2).

```
nazwa_zmiennej_plikowej >> zmienna_do_ktorej_zapisuje_dane;
plik >> dane;
```

2.2.2 Odczyt danych wierszami.

Do odczytu danych wierszami posłuży nam już znana z rozdziału 2 funkcja **getline()**.

```
getline( plik, dane);
```

Dane wczytane za pomocą funkcji getline() są zawsze traktowane jako tekst.

2.2.3 Odczyt danych blokami.

Do odczytu danych blokami wykorzystana zostanie funkcja **read()**. Dane wczytywane przez nią mogą być wczytane jako tekst, lub jako dane binarne (włączając tryb **ios::bin**).
W poniższym przykładzie możemy odczytać z pliku 1024 znaki do tablicy buforowej temp.

```
char temp[ 1024 ];
plik.read( temp, 1024 );
```

2.3 Zapis danych do pliku

Zapisywanie danych do pliku działa równie prosto co ich doczytywanie z pliku. Należy jednak pamiętać, że proces zapisu działa specyficznie. Możliwe jest dopisywanie danych, jeżeli znajdujemy się na końcu pliku, lub nadpisać dane, gdy jesteśmy w innym miejscu niż EOF (end of file).

2.3.1 Zapis danych za pomocą strumienia.

Ten sposób zapisu działa analogicznie do **std::cout«**, z tym, że tekst przesyłany jest do pliku, a nie na urządzenie peryferyjne.

```
nazwa_zmiennej_plikowej << zmienna_z_ktorej_zapisuje_dane;
plik << dane;
```

2.3.2 Zapis danych blokami.

Do zapisu danych blokami wykorzystana zostanie funkcja **write()**. W poniższym przykładzie zmienna tekst (typu string) pełni rolę bufora (&tekst[0] to stworzenie wskaźnika do pierwszego elementu zmiennej) natomiast tekst.length() mówi kompilatorowi, żeby skopiował ilość danych równą długości bufora tekst.

```
getline( cin, tekst );
plik.write( & tekst[ 0 ], tekst.length() );
```

Przykład:

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    fstream plik;
    plik.open( "plik.txt", ios::in | ios::out );
    if( plik.good() == true ) {
   cout << "Uzyskano dostep do pliku!" << endl;

        //tu nalezy przeprowadzac operacje na pliku (zapis/odczyt
           etc.)
        //zapis do pliku

        plik << "Ten tekst pojawi sie w naszym pliku" << endl;
        string napis, tekst;
        tekst="to jest string, ktory przeniesie sie do pliku\n";
        plik << tekst;
        cout<<"Podaj dowolny string, zostanie on zapisany w "
        <<" pliku"<<endl;
        getline(cin, napis);
        plik.write(&napis[0],napis.length());

        //odczyt z pliku

        //przeniesienie kursora o 3 bajty w strone poczatku pliku
        plik.seekg( + 3, std::ios_base::beg);

        string napis1, napis2;
        plik >> napis1;
        cout << "Pierwsza linijka pliku:" << endl << napis1 << endl;

        getline(plik, napis2);
        cout << "Kolejna linijka pliku: " << endl << napis2 << endl;

        char temp [10];
        plik.read(temp, 10);
        cout << "kolejna linijka pliku: " << endl;
        for(int i=0; i<10; i++) cout <<temp[i];
        cout << endl;
```

```
        plik.close();
    }
    else cout << "Brak dostepu do pliku " << endl;

    return( 0 );
}
```

Na potrzeby powyższego przykładu należy stworzyć w katalogu z plikiem *.cpp programu plik tekstowy: plik.txt.

Zadanie

W oparciu o zagadnienia wymienione w niniejszej instrukcji proszę udoskonalić prosty interface użytkowy automatu do sprzedaży przekąsek i napoi stworzony na poprzednich zajęciach.

Wymogi dotyczące programu:
1. Program wczytuje listę „produktów" wraz z ilością dostępnych sztuk, z zewnętrznego pliku tekstowego lub binarnego podczas uruchomienia, oraz za każdym razem, gdy lista jest odświeżana.
2. Po dokonaniu zakupu, liczba produktów jest odświeżana i zapisywana do pliku zewnętrznego.