Lista 7 Zad. 1

Pierwsza wersja programu

1. Przygotuj plik do wczytania: plik tekstowy z pojedynczą liczbą. Umieść go w tym samym folderze, co projekt (*.cpp).
2. Nazwa pliku
   name – nazwa zmiennej – może być dowolna (nazwę pliku użytkownik powinien podać z rozszerzeniem, np. zad.txt)

```
#include<iostream>
#include<string>

using namespace std;

int main(){
	string name;
	cout<<"Podaj nazwe pliku"<<endl;
	cin>>name;
	system("pause");
}
```

3. Otwieranie pliku do czytania – biblioteka + utworzenie zmiennej typu strumień wejściowy dla pliku

   ifstream - to typ zmiennej, tak jak int, double, vector<int>, string, char, itp…

   ist – nazwa zamiennej – może być dowolna

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

using namespace std;

int main(){
	string name;
	cout<<"Podaj nazwe pliku"<<endl;
	cin>>name;
	//otwieranie pliku do czytania
	ifstream ist(name.c_str());//zmienna ost typu strumień wejściowy dla pliku o
	nazwie name; otwiera plik o nazwie name do czytania
	system("pause");
    }
```

4. Wyrzuć wyjątek jeśli zmienna **ist** nie istnieje – bloki: try, catch i wyrzucanie wyjątku

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

using namespace std;

int main(){
	try{
		string name;
		cout<<"Podaj nazwe pliku"<<endl;
		cin>>name;
		//otwieranie pliku do czytania
		ifstream ist(name.c_str());//zmienna ost typu strumień wejściowy dla
pliku o nazwie name
                                                    //otwiera plik o nazwie name do
czytania
```

```
            if (!ist) throw runtime_error("Nie można otworzyć pliku wejsciowego!");
            system("pause");
        }
        catch(runtime_error &er){
        cerr<<er.what()<<endl;
        system("pause");}
    }
```

5. Wczytaj liczbę z pliku i ją wyświetl – to już znamy: ist to jak nasze wcześniejsze cin

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

using namespace std;

int main(){
        try{
            string name;
            cout<<"Podaj nazwe pliku"<<endl;
            cin>>name;
            //otwieranie pliku do czytania
            ifstream ist(name.c_str());//zmienna ost typu strumień wejściowy dla
pliku o nazwie name
                                                    //otwiera plik o nazwie name do
czytania
            if (!ist) throw runtime_error("Nie można otworzyć pliku wejsciowego!");
            //czytamy
            double a;
            ist>>a;
            cout<<a<<endl;
            system("pause");
        }
        catch(runtime_error &er){
        cerr<<er.what()<<endl;
        system("pause");}
    }
```

Druga wersja programu – czytaj kilka liczb i przechowuj w wektorze

1. Dodajemy bibliotekę vecotr, i wczytujemy serię liczb w pętli while – analogicznie jak robiliśmy to przy użyciu cin oraz wyświetlamy wektor

```
#include<iostream>
#include<string>
#include<fstream>
#include<vector>

using namespace std;

int main(){
        try{
            string name;
            cout<<"Podaj nazwe pliku"<<endl;
            cin>>name;
            //otwieranie pliku do czytania
            ifstream ist(name.c_str());//zmienna ost typu strumień wejściowy dla
pliku o nazwie name
                                                    //otwiera plik o nazwie name do
czytania
            if (!ist) throw runtime_error("Nie można otworzyć pliku wejsciowego!");
            //czytamy
```

```
            double a;
            vector<double>liczby;
            while(ist>>a)
                  liczby.push_back(a);
            for (int i=0; i<liczby.size(); i++) cout<<liczby[i]<<endl;
            system("pause");
      }
      catch(runtime_error &er){
      cerr<<er.what()<<endl;
      system("pause");}
   }
```

Trzecia wersja programu – pogrupujmy w funkcje

1. Ustalmy, co może być osobną funkcją: czytanie liczb z pliku do wektora, wyświetlanie wektora; Pamiętaj, że wywołując funkcje argument musi być tego samego typu, ale nie musi mieć tej samej nazwy, co w nagłówku funkcji.

```
#include<iostream>
#include<string>
#include<fstream>
#include<vector>

using namespace std;

vector<double> czytaj(istream &in){
      double a;
      vector<double>liczby;
      while(in>>a)
            liczby.push_back(a);
      return liczby;
}

void wyswietl(vector<double> &liczby){
      for (int i=0; i<liczby.size(); i++) cout<<liczby[i]<<endl;
}

int main(){
      try{
            string name;
            cout<<"Podaj nazwe pliku"<<endl;
            cin>>name;
            //otwieranie pliku do czytania
            ifstream ist(name.c_str());//zmienna ost typu strumień wejściowy dla
pliku o nazwie name
                                                //otwiera plik o nazwie name do
czytania
            if (!ist) throw runtime_error("Nie można otworzyć pliku wejsciowego!");
            //czytamy
            vector<double>pomiary=czytaj(ist);
            wyswietl(pomiary);
            system("pause");
      }
      catch(runtime_error &er){
      cerr<<er.what()<<endl;
      system("pause");}
   }
```

2. Chcemy, aby nasza funkcja do wyświetlania była bardziej uniwersalna, tzn. pozwalała nam wyświetlać wynik na konsoli, ale także np. zapisywać do pliku, powinna zatem móc korzystać z różnych strumieni wyjścia **ostream** (zmieniamy nazwę – nie trzeba – żeby nie mylić użytkownika

na wypisz). Parametry wywoływania takiej funkcji to wektor liczb oraz typ strumienia wyjścia – tutaj **cout**, ponieważ chcemy wyświetlić wynik na konsoli.

```
#include<iostream>
#include<string>
#include<fstream>
#include<vector>

using namespace std;

vector<double> czytaj(istream &in){
	double a;
	vector<double>liczby;
	while(in>>a)
		liczby.push_back(a);
	return liczby;
}

void wypisz(vector<double> &liczby, ostream &out){
	for (int i=0; i<liczby.size(); i++) out<<liczby[i]<<endl;
}

int main(){
	try{
		string name;
		cout<<"Podaj nazwe pliku"<<endl;
		cin>>name;
		//otwieranie pliku do czytania
		ifstream ist(name.c_str());//zmienna ost typu strumień wejściowy dla
pliku o nazwie name
                                                    //otwiera plik o nazwie name do
czytania
		if (!ist) throw runtime_error("Nie można otworzyć pliku wejsciowego!");
		//czytamy
		vector<double>pomiary=czytaj(ist);
		wypisz(pomiary,cout);
		system("pause");
	}
	catch(runtime_error &er){
	cerr<<er.what()<<endl;
	system("pause");}

   }
```

Czwarta wersja – użytkownik może przeczytać i może dopisać liczby na końcu pliku

1. Użytkownik ma wybór, czy chce przeczytać czy dopisać coś na końcu linii. A może warto pozwolić mu na obie te operacje dopóki nie zechce opuścić programu? Jasne, że warto. W tym celu możemy zastosować np. blok **switch – case** w pętli **while** z warunkiem wyjścia.

```
#include<iostream>
#include<string>
#include<fstream>
#include<vector>

using namespace std;

vector<double> czytaj(istream &in){
	double a;
	vector<double>liczby;
```

```
	while(in>>a)
		liczby.push_back(a);
	return liczby;
}

void wypisz(vector<double> &liczby, ostream &out){
	for (int i=0; i<liczby.size(); i++) out<<liczby[i]<<endl;
}

int main(){
	try{
		string name;
		cout<<"Podaj nazwe pliku"<<endl;
		cin>>name;
		//otwieranie pliku do czytania
		char dzialanie;
		cout<<"Co chcesz zrobic z plikiem: o - otworzyc, n - nadpisac, q - 
zakonczyc prace?\n";
		while(cin>>dzialanie && dzialanie!='q'){
			switch(dzialanie){
			case 'o':{
				ifstream ist(name.c_str());//zmienna ost typu strumień 
wejściowy dla pliku o nazwie name
								//otwiera plik o nazwie name do 
czytania
				if (!ist) throw runtime_error("Nie można otworzyć pliku 
wejsciowego!");
				//czytamy
				vector<double>pomiary=czytaj(ist);
				wypisz(pomiary,cout);
				break;
					}
			case 'n':{
				//nadpisz
				break;
					}
			case 'q':{
				//zakoncz
				break;
					}
			default: throw string("Niepoprawne dzialanie");
			}
		}
		system("pause");
	}
	catch(runtime_error &er){
		cerr<<er.what()<<endl;
		system("pause");}
	catch(string &er){
		cerr<<er<<endl;
		system("pause");
	}
}
```

2. Wprowadzamy dane do dopisania do pliku. Przy wczytywaniu liczb z pliku mogły wystąpić różne rodzaje błędów – zakończenie pętli while:
   - fail() – zakończenie poprzez podanie znaku kończącego lub błąd formatu (np. podano double zamiast int)
   - eof() – zakończenie poprzez znak końca pliku
   - bad() – poważny problem

Musimy zając się tymi błędami i wyczyścić strumień. W tym celu napiszmy dodatkową funkcję: wyczyść_strumien. Argumentami tej funkcji będą strumień wejścia (typ istream oraz znak końcowy – terminator - typ char, np.: '\n')

```
#include<iostream>
#include<string>
#include<fstream>
#include<vector>

using namespace std;

vector<double> czytaj(istream &in){
	double a;
	vector<double>liczby;
	while(in>>a)
		liczby.push_back(a);
	return liczby;
}

void wypisz(vector<double> &liczby, ostream &out){
	for (int i=0; i<liczby.size(); i++) out<<liczby[i]<<endl;
}

void wyczysc_strumien(istream &in, char terminator){
	if (in.bad()) throw runtime_error("Poważny problem ze standardowym wejsciem!");
	else if (in.fail() || in.eof()) {
	in.clear(); // czyscimy stan strumienia, bez tego nie mozemy z niego korzystac
	in.ignore(numeric_limits<streamsize>::max(),terminator); // pomijamy wszystkie
znaki az do \n lub konca pliku (eof)
	}
}

int main(){
	try{
		string name;
		cout<<"Podaj nazwe pliku"<<endl;
		cin>>name;
		char dzialanie;
		cout<<"Co chcesz zrobic z plikiem: o - otworzyc, n - nadpisac, q -
zakonczyc prace?\n";
		while(cin>>dzialanie && dzialanie!='q'){
			switch(dzialanie){
			case 'o':{
				ifstream ist(name.c_str());//zmienna ost typu strumień
wejściowy dla pliku o nazwie name
								//otwiera plik o nazwie name do
czytania
				if (!ist) throw runtime_error("Nie można otworzyć pliku
wejsciowego!");
				//czytamy
				vector<double>pomiary=czytaj(ist);
				wypisz(pomiary,cout);
				break;
					}
			case 'n':{
				cout<<"Podaj dane do wczytania do pliku\n";
				vector<double>dodatkowe_dane=czytaj(cin);
				wyczysc_strumien(cin,'\n');
				break;
					}
			case 'q':{
				//zakoncz
				break;
```

```
                                        }
                        default: throw string("Niepoprawne dzialanie");
                        }
                }
                system("pause");
        }
        catch(runtime_error &er){
                cerr<<er.what()<<endl;
                system("pause");}
        catch(string &er){
                cerr<<er<<endl;
                system("pause");
        }
}
```

3. Otwieramy plik do nadpisania **ofstream** – analogicznie jak ifstream i używając funkcji wypisz zapisujemy do tego strumienia wyjścia

```
#include<iostream>
#include<string>
#include<fstream>
#include<vector>

using namespace std;

vector<double> czytaj(istream &in){
        double a;
        vector<double>liczby;
        while(in>>a)
                liczby.push_back(a);
        return liczby;
}

void wypisz(vector<double> &liczby, ostream &out){
        for (int i=0; i<liczby.size(); i++) out<<liczby[i]<<endl;
}

void wyczysc_strumien(istream &in, char terminator){
        if (in.bad()) throw runtime_error("Poważny problem ze standardowym wejsciem!");
        else if (in.fail() || in.eof()) {
        in.clear(); // czyscimy stan strumienia, bez tego nie mozemy z niego korzystac
        in.ignore(numeric_limits<streamsize>::max(),terminator); // pomijamy wszystkie
znaki az do \n lub konca pliku (eof)
        }
}

int main(){
        try{
                string name;
                cout<<"Podaj nazwe pliku"<<endl;
                cin>>name;
                char dzialanie;
                cout<<"Co chcesz zrobic z plikiem: o - otworzyc, n - nadpisac, q -
zakonczyc prace?\n";
                while(cin>>dzialanie && dzialanie!='q'){
                        switch(dzialanie){
                        case 'o':{
                                ifstream ist(name.c_str());//zmienna ost typu strumień
wejściowy dla pliku o nazwie name
                                                        //otwiera plik o nazwie name do
czytania
                                if (!ist) throw runtime_error("Nie można otworzyć pliku
wejsciowego!");
                                //czytamy
```

```
                        vector<double>pomiary=czytaj(ist);
                        wypisz(pomiary,cout);
                        break;
                                }
                case 'n':{
                        cout<<"Podaj dane do wczytania do pliku\n";
                        vector<double>dodatkowe_dane=czytaj(cin);
                        wyczysc_strumien(cin,'\n');
                        ofstream ost(name.c_str());//otwieramy plik do nadpisania
                        if (!ost) throw runtime_error("Nie można otworzyć pliku
wejsciowego!");
                        wypisz(dodatkowe_dane,ost);
                        break;
                                }
                case 'q':{
                        //zakoncz
                        break;
                                }
                default: throw string("Niepoprawne dzialanie");
                }
        }
        system("pause");
    }
    catch(runtime_error &er){
        cerr<<er.what()<<endl;
        system("pause");}
    catch(string &er){
        cerr<<er<<endl;
        system("pause");
    }
}
```

Wersja piąta – dodajmy jeszcze opcje dopisania na końcu linii

1. Dodatkowa opcja w switch – case; podobnie jak w przypadku nadpisania, tylko teraz użyjmy parametru **iso_base::app** (więcej Tryby otwierania plików na wykładzie)

```
#include<iostream>
#include<string>
#include<fstream>
#include<vector>

using namespace std;

vector<double> czytaj(istream &in){
    double a;
    vector<double>liczby;
    while(in>>a)
        liczby.push_back(a);
    return liczby;
}

void wypisz(vector<double> &liczby, ostream &out){
    for (int i=0; i<liczby.size(); i++) out<<liczby[i]<<endl;
}

void wyczysc_strumien(istream &in, char terminator){
    if (in.bad()) throw runtime_error("Poważny problem ze standardowym wejsciem!");
    else if (in.fail() || in.eof()) {
    in.clear(); // czyscimy stan strumienia, bez tego nie mozemy z niego korzystac
    in.ignore(numeric_limits<streamsize>::max(),terminator); // pomijamy wszystkie
znaki az do \n lub konca pliku (eof)
    }
}
```

```
int main(){
	try{
		string name;
		cout<<"Podaj nazwe pliku"<<endl;
		cin>>name;
		char dzialanie;
		cout<<"Co chcesz zrobic z plikiem: o - otworzyc, n - nadpisac, d - dopisz
na końcu pliku, q - zakonczyc prace?\n";
		while(cin>>dzialanie && dzialanie!='q'){
			switch(dzialanie){
			case 'o':{
				ifstream ist(name.c_str());//zmienna ost typu strumień
wejściowy dla pliku o nazwie name
                                                    //otwiera plik o nazwie name do
czytania
				if (!ist) throw runtime_error("Nie można otworzyć pliku
wejsciowego!");
				//czytamy
				vector<double>pomiary=czytaj(ist);
				wypisz(pomiary,cout);
				break;
					}
			case 'n':{
				cout<<"Podaj dane do wczytania do pliku\n";
				vector<double>dodatkowe_dane=czytaj(cin);
				wyczysc_strumien(cin,'\n');
				ofstream ost(name.c_str());//otwieramy plik do nadpisania
				if (!ost) throw runtime_error("Nie można otworzyć pliku
wejsciowego!");
				wypisz(dodatkowe_dane,ost);
				break;
					}
			case 'd':{
				cout<<"Podaj dane do wczytania do pliku\n";
				vector<double>dodatkowe_dane=czytaj(cin);
				wyczysc_strumien(cin,'\n');
				ofstream ost(name.c_str(),ios_base::app);//otwieramy plik
do dopisania na końcu linii
				if (!ost) throw runtime_error("Nie można otworzyć pliku
wejsciowego!");
				wypisz(dodatkowe_dane,ost);
				break;
					}
			case 'q':{
				//zakoncz
				break;
					}
			default: throw string("Niepoprawne dzialanie");
			}
		}
		system("pause");
	}
	catch(runtime_error &er){
		cerr<<er.what()<<endl;
		system("pause");}
	catch(string &er){
		cerr<<er<<endl;
		system("pause");
	}
   }
```