



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

PODSTAWY INFORMATYKI

Informatyka Stosowana – ROK I

Laboratoria – zajęcia nr 1

mgr inż. Krzysztof Bzowski



Kilka informacji

Krzysztof Bzowski

kbzowski@agh.edu.pl

Tel. +12 6172615

Jabber: kbzowski@agh.edu.pl

B5 / 605

<http://home.agh.edu.pl/~kbzowski/>

- Konsultacje
 - Poniedziałek: 6:30 – 8:00
 - Inne dni (umówić się mailowo)

Zasady zaliczenia

- Egzamin
- **3 kolokwia (~1 godzinne)**
- Wejściówki – jedno minutowe trywialne pytanko
- Aktywność na zajęciach (0.1 - 0.5pkt)
- Ocena końcowa:

$$O_{wejciówka}^{\max} = 10 \cdot 1 = 10$$

$$O_{kolokwium}^{\max} = 15 \cdot 3 = 45$$

$$O_{końcowa} = 0.6 \cdot O_{wejciówka} + 0.4 \cdot O_{kolokwium} = 24^{\max}$$

$$0.6 \cdot 10 + 0.4 \cdot 16 = 12.1$$

Plan

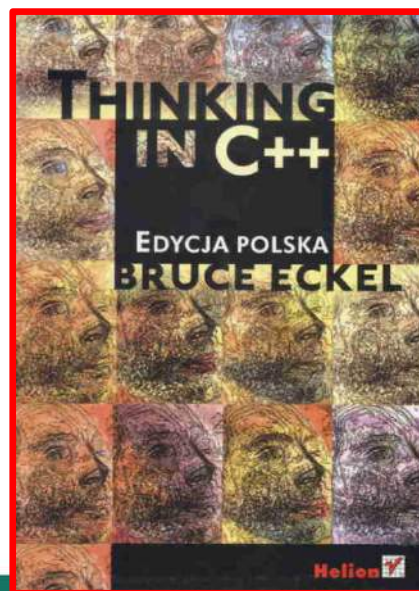
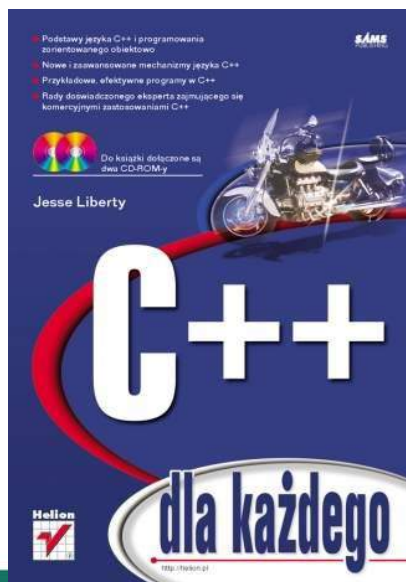
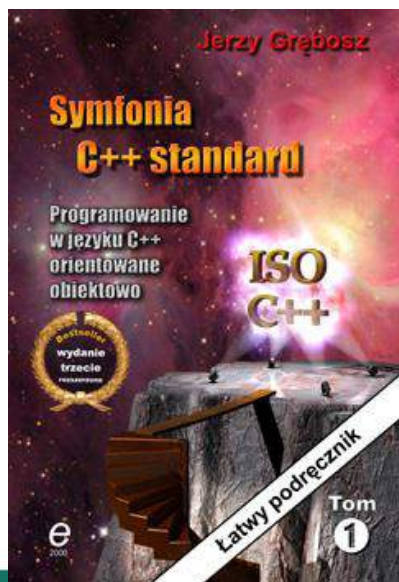
1	Wstęp, Środowisko programistyczne, budowa programu, zmienne, stałe, nazewnictwo, cin, cout, operatory matematyczne (wstęp)
2	Pętle (for, while, do while), operatory, instrukcje warunkowe, Debugging
3	Funkcje (co jak gdzie po co, argumenty, argumenty domyślne, wartość a referencja, przeładowanie nazw funkcji)
4	Tablice statyczne 1D i 2D (tworzenie, uzupełniania, przeglądanie)
5	KOŁOKWIUM I
6	Łącuchy znaków i operacje na łańcuchach
7	Wskaźniki, rzutowanie typów
8	Dynamiczna alokacja pamięci dla zmiennych i tablic
9	Argumenty funkcji main
10	KOŁOKWIUM II
11	Struktury I (idea, deklaracja i definicja)
12	Stringi i wektory
13	Operacja na plikach
14	Pomiar czasu wykonywania operacji, programy wieloplikowe, tworzenie bibliotek
15	KOŁOKWIUM III

Kolokwia

- W czasie semestru odbędą się trzy kolokwia.
- **Niezaliczone kolokwia nie będą poprawiane w trakcie semestru.**
- **Nieusprawiedliwiona nieobecność na kolokwium jest równoznaczna z otrzymaniem ZERU PUNKTÓW.**
- Student, który usprawiedliwi swoją nieobecność na kolokwium może je zaliczać w terminie podanym przez prowadzącego.
- Ocena końcowa jest pozytywna (3.0), jeżeli student uzyska ponad 50% punktów (to jest ponad 12.2).
- Student, który uzyskał ocenę średnią niższą niż 3.0 może w czasie sesji dwukrotnie przystąpić kolokwium poprawkowego (kolokwium poprawkowe obejmuje materiał z całego semestru), pod warunkiem, że uczęszczał na zajęcia.

Literatura

- Każda książka o programowaniu w c++ (zakres programowania proceduralnego + struktury)
- Na początek:
 - Jerzy Grębosz - Symfonia C++
- Google!



<http://pl.wikibooks.org/wiki/C++>

- Linux: gcc 4.7 – 4.8
- Windows: Visual Studio 2008/2010/2012
 - Darmowy Visual Studio 2012 Express do pobrania:

<http://www.microsoft.com/visualstudio/eng/downloads#d-2012-express>



Polecam wersję angielską!



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski



Kartkówka

Napisać pętlę *for* która 5 razy wypisze Twoje imię i nazwisko w nowej linii.

(bez deklaracji funkcji, #include, etc.)

Zadanie 1

- Napisz program proszący użytkownika o podanie dwóch liczb rzeczywistych **a** i **b**, a następnie wyświetlający wynik:
 - Dodawania a i b
 - Różnicy a i b
 - Iloczynu a i b
 - Ilorazu a przez b

Założmy, że użytkownik nie poda cyfry 0

Zadanie 2

- Narysować kwadrat i trójkąt (wypełnione w środku i puste) o zadanej długości boku/wysokości. Wykorzystać pętlę i instrukcje warunkowe

```
* * * * *  
*       *  
*       *  
*       *  
* * * * *
```

```
*  
* *  
*  *  
*   *  
* * * * *
```



ZADANIE 3

Narysuj choinkę o zadanej wysokości.
Choinka ma być wypełniona w środku i pusta.

np. wysokosc = 4

```
  *
 ***
*****
*****
```

```
  *
 * *
  _
 *   *
  _
*****
```



Zadanie 3*

Narysować sześciokąt i ośmiokąt z gwiazdek (*)

ZADANIE 4

Wypisać n wyrazów ciągu Fibonacciego.

$$F_n := \begin{cases} 0 & \text{dla } n = 0; \\ 1 & \text{dla } n = 1; \\ F_{n-1} + F_{n-2} & \text{dla } n > 1. \end{cases}$$

np.

0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	
---	---	---	---	---	---	---	----	----	----	----	----	-----	-----	-----	-----	-----	------	------	--



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski

ZADANIE 1

Napisz funkcję *liczby_pitagorejskie()*, która ma trzy parametry formalne *a*, *b*, *c*, będące liczbami całkowitymi (int). Funkcja zwraca wartość jeden (1), jeśli zadane liczby są liczbami pitagorejskimi oraz zero (0) w przeciwnym wypadku. Liczby pitagorejskie spełniają warunek:

$$a \cdot a + b \cdot b = c \cdot c$$



ZADANIE 2

Napisz funkcję, która dostaje jako argumenty pięć liczb typu unsigned int i zwraca jako wartość sumę podanych liczb. Funkcję napisz w taki sposób, żeby liczyła sumę także dwóch, trzech i czterech argumentów.

ZADANIE 3

Napisz funkcję, która stwierdza, czy zadana jako parametr liczba całkowita jest liczbą pierwszą. Wartością funkcji ma być *prawda*, jeśli liczba spełnia warunek oraz *fałsz* w przeciwnym wypadku.

ZADANIE 4

Napisz funkcję *swap()*, która wymienia wartościami dwie zadane liczby zmiennoprzecinkowe.

Po wyjściu funkcji argumenty przekazane w postaci argumentów liczby powinny pozostać zamienione.

ZADANIE 5

Napisać funkcję, która jako argument przyjmuje liczbę całkowitą.

Funkcja wypisze na ekran wszystkie liczby 5-cio cyfrowe, których suma cyfr jest równa podanej liczbie.



ZADANIE 6

Napisz funkcję, która zwróci sumę cyfr w podanej liczbie(int).

$$5921=5+9+2+1=17$$

ZADANIE 7

Napisz funkcję, która sprawdzi czy przekazana do niej liczba jest liczbą palindromiczną.

Liczba palindromiczna to liczba, która przy czytaniu z lewej strony do prawej i odwrotnie jest jednakowa.

Liczby takie nazywane są także liczbami symetrycznymi. Przykłady takich liczb to: 7, 57775, 626, 1111111...



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski

ZADANIE 1

Utworzyć tablicę 20 elementową typu `int` i wypełnić ją liczbami `<30,49>` w odwrotnej kolejności.

Program powinien składać się z funkcji wypełniającej tablicę liczbami oraz z funkcji wypisującej zawartość tablicy na ekran

Jak wygenerować liczbę losową (int)?

```
#include <time.h>
```

```
// Wygeneruj liczbę pseudolosowa z zakresu <min, max)
```

```
int losowa(const int min, const int max){
```

```
    static bool first = false;
```

```
    if(!first){
```

```
        srand(time(0));
```

```
        // srand inicjalizacja - powinna byc uzyta tylko
```

```
        // raz na cale dzialanie programu!
```

```
        first = true;
```

```
    }
```

```
    int los = min + (rand() % (int)(max - min + 1));
```

```
    return los;
```

```
}
```

Jak wygenerować liczbę losową (float)?

```
#include <time.h>
```

```
float losowa(const float min, const float max){  
    static bool first = false;  
    if(!first){  
        srand(time(0));  
        // srand inicjalizacja - powinna byc uzyta tylko  
        // raz na cale dzialanie programu!  
        first = true;  
    }  
  
    float los = min + (float)rand()/((float)RAND_MAX/(max-min));  
    return los;  
}
```

ZADANIE 2

Uzupełnić dowolną tablicę typu float losowymi danymi. Znaleźć element największy i najmniejszy. Wypisać tablicę na ekran.

ZADANIE 3

Napisz funkcję, która otrzymuje dwa argumenty: dodatnią liczbę całkowitą n oraz n -elementową tablicę **tab** o elementach typu **int** i zwraca jako wartość średnią arytmetyczną elementów tablicy **tab**. Dla testu tablicę należy wypełnić liczbami losowymi.

ZADANIE 4

Wylosować 100 liczb z zakresu $[0,9]$ i zapisać je w tablicy. Zliczyć wystąpienie każdej z nich. Wynik zapisać w tablicy.

PRZYKŁAD

3	2	0	0	1	8	9	5	1	7	5	4	7	8	7	5	1	0	2
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

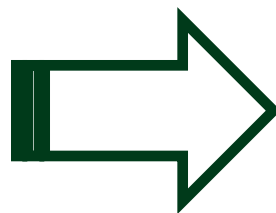
Indeksy	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
Wartości	3	3	2	1	1	3	0	3	2	1

ZADANIE 5

Stwórz macierz 4 x 6, wypełnij ją losowymi danymi, napisz program który dokona jej transpozycji.

PRZYKŁAD 4x4

5	10	6	1
8	4	5	0
0	7	5	4
5	9	4	1



5	8	0	5
10	4	7	9
6	5	5	4
1	0	4	1

ZADANIE 6

Zaimplementować mnożenie dowolnej macierzy i wektora o rozmiarach znanych w etapie kompilacji.

TEORIA

$$Ax = b$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \end{Bmatrix}$$

ZADANIE 6

Napisać funkcję, która zamieni liczbę (int) w zapisie dziesiętnym na binarną a wynik przechowa w tablicy.

$$25_{10} = 11001_2$$



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski

ZADANIE

Poniżej zdefiniowany jest pewien ciąg, którego kolejne wyrazy generowane są w sposób rekurencyjny:

$$a_n = \begin{cases} -1 & \text{dla } n = 0 \\ -a_{n-1} \cdot n - 3 & \text{dla } n > 0 \end{cases}$$

Napisz program, który znajdzie wartość n -tego wyrazu ciągu i wypisze ją na ekran. Wykorzystaj rekurencje.

ZADANIE

Wypisać n -ty wyraz ciągu Fibonacciego z wykorzystaniem rekurencji.

$$F_n := \begin{cases} 0 & \text{dla } n = 0; \\ 1 & \text{dla } n = 1; \\ F_{n-1} + F_{n-2} & \text{dla } n > 1. \end{cases}$$

np.

0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	
---	---	---	---	---	---	---	----	----	----	----	----	-----	-----	-----	-----	-----	------	------	--

ZADANIE

Rekurencyjnie obliczyć sumę n pierwszych wyrazów ciągu:

$$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

ZADANIE

Wykorzystując własności łańcuchów znaków zbadać ich wielkość. Napisz funkcję, która zwróci rzeczywistą ilość znaków w łańcuchu znaków.

ZADANIE

Napisać program, który zapyta użytkownika o imię i nazwisko i zapisze je w tablicy.

ZADANIE

Napisz funkcję *sklej* otrzymującą jako argumenty trzy tablice znaków i zapisującą do trzeciej tablicy konkatencję napisów znajdujących się w dwóch pierwszych tablicach. Zakładamy, że w trzeciej tablicy jest wystarczająco dużo miejsca.

PRZYKŁAD

```
sklej(char* napis1, char* napis2, char* napis3)
```

```
char t1[] = "Ala ";  
char t2[] = "ma kota";
```



```
char t3[] = "Ala ma kota";
```

ZADANIE

Napisz funkcję *wytnij*, która dostaje jako argumenty napis oraz dwie liczby całkowite n i m , i wycina z otrzymanego napisu znaki o indeksach od n do m ($n \leq m$). Otrzymany w argumencie napis może mieć dowolną liczbę znaków (w tym mniejszą od n lub m). Konieczne jest sprawdzenie wartości argumentów.



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski

KARTKÓWKA

- Stwórz 10-elementową tablicę typu int.
- Stwórz wskaźnik do 5 elementu tej tablicy
- Za pomocą wskaźnika przypisz do tego elementu wartość równą 0
- Wypisz na ekran wartość tego elementu za pomocą wskaźnika

```
int tab[10];  
int* wsk = &tab[5];  
*wsk = 0;  
std::cout<<*wsk;
```

Podsumowanie: Dynamiczne tworzenie i usuwanie tablic

Zmienna

```
int *w = new int;  
*w = 10;
```

Zmienna

```
delete w;
```

Tablica

```
int *w = new int[n];
```

Tablica

```
delete [] w;
```

Tablica 2D

```
int **w = new int*[n];  
for(int i = 0; i < n; i++){  
    w[i] = new int[m];  
}
```

Tablica 2D

```
for(int i=0; i<n; i++){  
    delete [] w[i];  
}  
delete [] w;
```

Podsumowanie: Wskaźniki na funkcje

```
int funkcja (int a){  
    return a;  
}
```

```
int main(){  
    int (*pfun)(int a);  
    pfun = funkcja;  
    int wynik = pfun(5);  
}
```

ZADANIE 1

Napisz funkcję pozwalającą zwiększyć lub zmniejszyć rozmiar dowolnej tablicy liczb rzeczywistych. Zademonstruj jej działanie na dowolnym przykładzie.

ARGUMENTY FUNKCJI

- wskaźnik do tablicy
- rozmiar tablicy (**int** *staryRozmiar*)
- nowy rozmiar (**int** *nowyRozmiar*)

WARTOŚĆ ZWRACANA

Funkcja powinna zwracać wskaźnik do nowej tablicy.

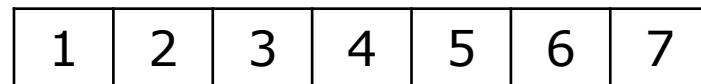
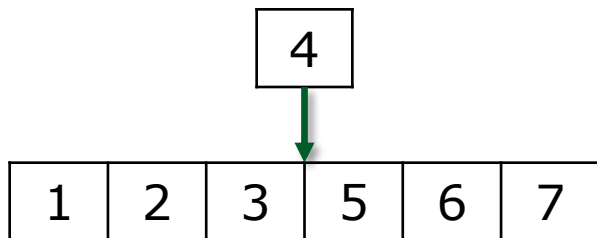
PRZYPADKI

Jeśli *staryRozmiar* < *nowyRozmiar* – uzupełnić zerami wolne miejsce

Jeśli *staryRozmiar* > *nowyRozmiar* – uciąć pozostałe elementy

ZADANIE 2

Napisz funkcję pozwalającą dodać w dowolne miejsce tablicy nowy element. Założyć że dane wprowadzone przez użytkownika są poprawne



ARGUMENTY FUNKCJI

- wskaźnik do tablicy (**float** * **array**)
 - wielkość tablicy (**int** **size**)
- pozycja do umieszczenia nowej wartości (**int** **position**)
 - nowa wartość (**float** **value**)

WARTOŚĆ ZWRACANA

funkcja powinna zwracać
wskaźnik do nowej tablicy

ZADANIE 3

Napisz funkcję pozwalającą dokonać sumowania dwóch dowolnych tablic dwuwymiarowych (macierzy). Funkcja powinna zwrócić wskaźnik do macierzy wynikowej. Pokaż na przykładzie jej działanie

ARGUMENTY FUNKCJI

- wskaźnik do macierzy A (**float** ** **matrixA**)
- wskaźnik do macierzy B (**float** ** **matrixB**)
 - ilość kolumn (**int** **width**)
 - ilość wierszy (**int** **height**)

WARTOŚĆ ZWRACANA

Funkcja powinna zwracać wskaźnik do nowej macierzy.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & a_{13} + b_{13} & a_{14} + b_{14} \\ a_{21} + b_{21} & a_{22} + b_{22} & a_{23} + b_{23} & a_{24} + b_{24} \\ a_{31} + b_{31} & a_{32} + b_{32} & a_{33} + b_{33} & a_{34} + b_{34} \\ a_{41} + b_{41} & a_{42} + b_{33} & a_{43} + b_{43} & a_{44} + b_{44} \end{bmatrix}$$

ZADANIE 4

Napisz funkcję, która otrzymuje trzy argumenty:

- wskaźnik na funkcję przyjmującą dwa argumenty typu **int** zwracające losową wartość typu **int**,

- wskaźnik na funkcję przyjmującą jeden argument typu **char*** (łańcuch znaków)

Zaimplementuj funkcję losującą liczbę z przedziału $\langle \text{min}, \text{max} \rangle$ oraz funkcję wyświetlającą komunikat w dowolny sposób.

Wykorzystaj obie zaimplementowane funkcje jako argumenty pierwszej funkcji.



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski

Argumenty z linii wywołania programu

```
int main(int argc, char *argv[])
{
    cout<<"Dostalem parametrow: "<<argc<<endl<<endl;
    for (int i=0; i<argc; i++){
        cout<<i<<"parametr to\t"<<argv[i]<<endl;
    }

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

ZADANIE 1

Napisz program przyjmujący z wiersza polecen dowolną ilość liczb rzeczywistych i liczący ich średnią.

ZAMIANA CIĄGU ZNAKÓW NA LICZBĘ

```
char* litera = "52.025";  
float liczba_rz = atof(litera);  
int liczba_ca = atoi(litera);
```

ZADANIE 2

ROT13

ROT13 - prosty szyfr przesuwający, którego działanie polega na zamianie każdego znaku alfabetu łacińskiego na znak występujący 13 pozycji po nim, przy czym wielkość liter nie ma przy przekształcaniu znaczenia. ROT13 jest przykładem szyfru cezara, opracowanego w Starożytnym Rzymie.

Napisz program przyjmujący dowolny ciąg znaków. Program ma pozwolić zaszyfrować i odszyfrować ten ciąg znaków szyfrem ROT13 a następnie wynik wypisze na ekran.

ABCDEFGHIJKLMNOPQRSTUVWXYZ



ROT13

NOPQRSTUVWXYZABCDEFGHIJLM



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski

Struktury - podsumowanie

```
struct Test{  
    int a;  
};
```



```
int main(){  
    Test obiektTest;  
    obiektTest.a = 10;  
    cout<<obektTest.a;  
}
```

```
struct Test{  
    int a;  
    float b;  
    char z;  
};
```

```
int main(){  
    Test test = {1, 2.56, 'c'};  
}
```

Struktury – funkcje składowe

```
struct Test {  
    int* wsk;  
    int rozmiar;  
  
    void stworzTab(){  
        wsk = new int[rozmiar];  
        for(int i = 0; i<rozmiar; i++)  
            wsk[i] = 0;  
    }  
  
    void sprzataj(){  
        delete [] wsk;  
    }  
};  
  
int main(){  
    Test obiektTest;  
    obiektTest.rozmiar = 10;  
    obiektTest.stworzTab();  
    obiektTest.sprzataj();  
}
```

Funkcje składowe cd

```
struct Test{  
    void funkcjaWew(){  
        cout<<"Jestem funkcja zdefiniowana w strukturze"<<endl;  
    }  
    void funkcjaZew();  
};
```

```
[ void Test::funkcjaZew(){  
    cout<<"Jestem funkcja zdefiniowana poza struktura"<<endl;  
}]
```

```
int main(){  
    Test obiektTestowy;  
    obiektTestowy.funkcjaWew();  
    obiektTestowy.funkcjaZew();  
}
```


Tablice obiektów

```
struct Test{  
    int a;  
};
```

```
int main(){  
    Test tab[10];  
    Test* dynTab = new Test[10];  
    for(int i = 0; i<10; i++){  
        tab[i].a = i;  
        dynTab[i].a = i;  
    }  
    delete [] dynTab;  
}
```

Wskaźnik do obiektu struktury

```
struct Test{  
    int a;  
};  
  
int main(){  
    Test* wskTest = new Test;  
    wskTest->a = 1;  
    (*wskTest).a = 1;  
    cout<<(*wskTest).a<<endl;  
    cout<<wskTest->a<<endl;  
    delete wskTest;  
}
```

Tablice wskaźników na obiekty

```
void testTablicyWskaznikow(){  
    Test** tabWskTest = new Test*[10];  
    for(int i = 0; i<10; ++i){  
        tabWskTest[i] = new Test;  
    }  
  
    tabWskTest[0]->a = 10;  
    cout<<tabWskTest[0]->a<<endl;  
  
    for(int i = 0; i<10; ++i){  
        delete tabWskTest[i];  
    }  
  
    delete[] tabWskTest;  
}
```

Struktura jako argument funkcji

```
struct Liczba{  
    int x;  
    int y;  
};
```

```
Liczba dodajLiczby(Liczba a, Liczba b){  
    Liczba wynik;  
    wynik.x = a.x + b.x;  
    wynik.y = a.y + b.y;  
    return wynik;  
}
```

Argument jako referencja

```
void FunkcjaZArgumentemJakoReferencja(Test& arg){  
    arg.a = 10;  
}  
void FunkcjaZArgumentemJakoWartosc(Test arg){  
    arg.a = 20;  
}  
int main(){  
    Test test;  
    test.a = 0;  
  
    FunkcjaZArgumentemJakoReferencja(test);  
    cout<<test.a;           // 10  
  
    FunkcjaZArgumentemJakoWartosc(test);  
    cout<<test.a;           // 10  
}
```

Statyczne składowe struktury

```
struct Test{  
    static int statyczna;  
    const static int stalaStatyczna = 1;  
};  
  
int Test::statyczna = 0;  
  
int main(int g, char* d[]){  
    Test::statyczna++;           // 1  
    Test obj1;  
  
    cout<<obj1.statyczna<<endl;  
    obj1.statyczna++;           // 2  
  
    Test obj2;  
    cout<<obj2.statyczna<<endl;  
}
```

```
struct Test{  
    static void wypisz(){  
        cout<<"XXX";  
    }  
};  
  
int main(int g, char* d[]){  
    Test::wypisz();  
}
```

ZADANIE 1A

Zdefiniuj strukturę **Trojkat** przechowującą długości boków trójkąta. Napisz funkcję pozwalającą obliczyć obwód trójkąta.

Zaproponuj rozwiązanie w postaci:

- funkcji składowej struktury **Trojkat**
- funkcji otrzymującej jako argument obiekt **Trojkat**
- funkcji otrzymującej jako argument wskaźnik do obiektu **Trojkat**

ZADANIE 1B

Zdefiniuj strukturę **LiczbaZespolona** służącą do przechowywania liczb zespolonych. Zdefiniowana struktura powinna zawierać pola **b** i **a** typu **float** służące do przechowywania odpowiednio części urojonej i rzeczywistej liczby zespolonej. Napisz funkcję składową, która wyświetli na ekran wartość liczby w poprawnym formacie (a+bi)

Napisz funkcję **dodaj**, pozwalającą zsumować dwie dowolne liczby zespolone.

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

Zaproponuj rozwiązanie jako:

- funkcję składową struktury **LiczbaZespolona**
- funkcję otrzymującą dwa obiekty typu **LiczbaZespolona** jako argumenty
- funkcję otrzymującą jako argument dwa wskaźniki do obiektów typu **LiczbaZespolona**

ZADANIE 2

Rozwiązanie:

<http://wklej.to/fINB8/text>

Zdefiniuj strukturę **Punkt** służącą do przechowywania współrzędnych punktów w trójwymiarowej przestrzeni kartezjańskiej (x, y, z)

Napisz funkcję, która otrzymuje jako argumenty tablicę o argumentach typu **Punkt** oraz jej rozmiar i zwraca jako wartość najmniejszą spośród odległości pomiędzy każdymi dwoma punktami przechowywanymi w tablicy `tab`. Zakładamy, że otrzymana w argumencie tablica ma co najmniej dwa argumenty.

Odległość w przestrzeni pomiędzy dwoma punktami definiujemy jako:

$$d(A, B) = \sqrt{(x_{1A} - x_{1B})^2 + (x_{2A} - x_{2B})^2 + (x_{3A} - x_{3B})^2}$$

ZADANIE 3

Stwórz strukturę **Parking** zawierającą tablicę, której elementy są obiektami struktur **Miejsce**. Struktura **Miejsce** przechowuje wartość typu bool z informacją czy jest zajęte czy nie, oraz wskaźnik do obiektu **Samochod** (który może to miejsce zajmować). Struktura **Samochod** powinna przechowywać markę (char*) i numer rejestracyjny (char*)

Utwórz funkcje odpowiedzialne za zarządzanie parkingiem:

- Dodawanie samochodów na parking (na określonym miejscu lub pierwszym wolnym),
- Usuwanie samochodów z parkingu z dowolnego miejsca,
- Wypisywanie samochodów obecnych na parkingu,
- Informowaniu o ilości wolnych miejsc.

Program powinien uwzględniać, że na parkingu może pomieścić określoną ilość samochodów (np. 50)

Propozycje dodatkowych funkcjonalności:

1. Miejsce na którym może parkować określony samochód (porównywanie tablic rejestracyjnych)
2. Blokada parkowania na określonych miejscach
3. Rozbudowa parkingu (zwiększenie ilości dostępnych miejsc)
4. Inne...








AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski

Lektora do poduszki

- Opis klasy vector z przykładami 
<http://pl.wikibooks.org/wiki/C++/Vector>
- Opis klasy string z przykładami 
<http://pl.wikibooks.org/wiki/C++/String>
- Iteratory z przykładami 
<http://pl.wikibooks.org/wiki/C++/Iteratory>
- Funkcje składowe klasy string z przykładami 
<http://www.cplusplus.com/reference/string/string/>
- Funkcje składowe klasy vector z przykładami 
<http://www.cplusplus.com/reference/vector/vector/>

String

INICJALIZACJA

```
#include <string>
using namespace std;
[...]
string napis;
napis = "text";
```

OPERATORY

Kopiowanie

`a = b`

Porównanie

`a == b`

Dodawanie do końca

`a += b`

empty()	Zwraca wartość true jeżeli napis jest pusty.
size(), length()	Zwraca ilość znaków w napisie.
at()	Zwraca znak o podanym położeniu, tak jak operator [], z tym że ta metoda jest bezpieczniejsza - wyrzuca wyjątek w przypadku wyjścia poza zakres stringa.
clear()	Usuwa wszystkie znaki z napisu.
erase(index, dlugosc)	Usuwa wybrane znaki.
find(podciag)	Znajduje podciąg w ciągu. Funkcja zwraca pozycje, na którym znaleziono podciąg lub std::string::npos.
swap(a,b)	Zamienia miejscami dwa stringi, a staje się b, a b staje się a.
substr(indeks, dlugosc)	Zwraca podciąg na podstawie indeksu początkowego i długości podciągu.
append(napis)	Dodaje zadany napis na końcu istniejącego ciągu.
c_str()	Zwraca napis w stylu języka C (stały wskaźnik typu const char*).

ZADANIE 1

Napisz funkcję która zweryfikuje, czy podany numer dowodu osobistego (jako String) jest prawidłowy.

Numery dowodu osobistego składają się z trzech liter oraz 6 cyfr.

```
char znak = '1';  
if(isdigit(znak)){  
    cout<<"Jest liczba";  
} else {  
    cout<<"Nie jest liczba";  
}
```

```
char znak = 'a';  
if(isalpha(znak)){  
    cout<<"Jest litera";  
} else {  
    cout<<"Nie jest litera";  
}
```

ZADANIE 2

Napisz funkcję która sprawdzi czy w dowolnym zdaniu występują oba podane wyrazy

ZADANIE 3

Napisz funkcję, która odwróci dowolny napis (string).

Np.

"DOWOLNY" -> "YNLOWOD"



std::vector<T> - podstawy

```
#include <vector>
using namespace std;
[...]

//vector<T> nazwaTablicy;
vector<int> vec;
vec.push_back(1);
vec.push_back(5);
vec.push_back(10);
vec.push_back(-2);
vec.push_back(4);

for(int i = 0; i<vec.size(); i++){
    cout<<vec[i]<<endl;
}
```

```
struct Object{
    int x;
};

int main(){
    vector<Object> wObj;
    for(int i = 0; i<4; i++){
        Object objDoWstawienia = {i};
        wObj.push_back(objDoWstawienia);
    }
}
```



std::vector<T> - Iteratory

```
vector<int> tab;
```

// inicjujemy wektor kolejnymi liczbami naturalnymi

```
tab.push_back(1);
```

```
tab.push_back(2);
```

```
tab.push_back(3);
```

*// **vector.begin()** – zwraca wskaźnik/iterator do pierwszego elementu wektora*

*// **vector.end()** – zwraca wskaźnik/iterator do elementu będącego **ZA** ostatnim elementem w wektorze*

*// **vector.back()** – zwraca referencje do ostatniego elementu w wektorze*

```
vector<int>::iterator i;
```

```
vector<int>::iterator start = tab.begin();
```

```
vector<int>::iterator koniec = tab.end();
```

```
for( i = start; i != koniec; ++i ) {
```

```
    cout<< *i <<endl;
```

```
}
```

std::vector<T> - Funkcje składowe

void swap(vector<T>& vec)	zamienia zawartości dwóch wektorów miejscami
void push_back(const T obj)	dodaje na końcu wektora kopię przekazanego argumentu
void pop_back()	usuwa ostatni element z wektora
void clear()	usuwa wszystkie elementy z wektora
void assign(size_t n, const T obj)	czyści wektor i wypełnia go n kopiami argumentu obj
void assign(iterator poczatek, iterator koniec)	czyści wektor i wypełnia go elementami z innego wektora z przedziału <poczatek;koniec>
iterator insert(iterator pos, T obj)	wstawia element obj przed wskazywaną przez iterator pos pozycją i zwraca iterator do dostawionego elementu
void insert(iterator pos, size_t n, const T obj)	wstawia n kopii argumentu obj przed pozycją wskazywaną przez iterator pos
void insert(iterator pos, iterator poczatek, iterator koniec)	wstawia przed pozycją wskazywaną przez iterator pos elementy między iteratorami poczatek i koniec (włącznie)
iterator erase(iterator pos)	usuwa element wskazywany przez pos i zwraca iterator do następnego elementu
iterator erase(iterator poczatek, iterator koniec)	usuwa elementy z przedziału <poczatek;koniec> i zwraca iterator do elementu za nimi

std::vector<T> - Przykłady

- Wstawianie wartości na pozycję nr 2

```
vector<int> vec;  
tab.push_back(1);  
tab.push_back(1);  
tab.push_back(1); // 1, 1, 1  
vector<int>::iterator it = vec.begin();  
vec.insert(it+2, 200); // 1, 1, 200, 1
```

- Sortowanie wektora/tablicy

```
#include <algorithm>  
[...]  
int tablica[] = {32,71,12,45,26,80,53,33};  
vector<int> vec(tablica, tablica+8); // kopiowanie tablicy do wektora  
sort(vec.begin(), vec.end()); // 12,26,32,33,45,53,71,80  
// można też sortować część wektora!  
sort(vec.begin(), vec.begin()+4); // 12,32,45,71,26,80,53,33
```

std::vector<T> - Przykłady

- Sortowanie przy użyciu funkcji porównującej (komparatora)

// Funkcja porównująca dwie wartości/obiekty – używana do sortowania
// Można ją wykorzystać do sortowania wektora obiektów ze względu na konkretną zmienną składową!

```
bool comparer (int i, int j) {  
    if(i>j) return true;    // Pierwsza wartość większa od drugiej – gwarantuje sortowanie malejące  
    return false;  
}
```

```
int main(){  
    int tablica[] = {32,71,12,45,26,80,53,33};  
    std::vector<int> vec(tablica, tablica+8); // kopiowanie tablicy do wektora  
    std::sort(vec.begin(), vec.end(), comparer); // 80,71,53,45,33,32,26,12  
}
```

std::vector<T> - Przykłady

- Wyszukiwanie obiektów w wektorze

```
#include <algorithm>
```

```
#include <vector>
```

```
[...]
```

```
vector<int> vec;
```

```
vec.push_back(10);
```

```
vec.push_back(20);
```

```
vec.push_back(15);
```

```
int szukana = 15;
```

```
// find(iterator do początku przedziału, iterator do końca przedziału, szukana wartosc)
```

```
vector<int>::iterator iter = find(vec.begin(), vec.end(), szukana);
```

```
if(iter != vec.end()){
```

```
// różnica między położeniami iteratora początku i znalezionego
```

```
// czyli indeks elementu w wektorze
```

```
int index = distance(vec.begin(), iter);
```

```
cout<<szukana<<" na pozycji "<<index;
```

```
}
```

ZADANIE 4

Stwórz strukturę typu **LiczbaZespolona**.

Utwórz wektor zawierający kilka obiektów typu **LiczbaZespolona**. Posortuj wektor malejąco ze względu na część rzeczywistą liczby zespolonej.

ZADANIE 5

Napisz program, który będzie gromadził w kontenerze vector wpisane z klawiatury imiona. Po wpisaniu słowa "koniec" zostanie wypisane najdłuższe imię.

ZADANIE 6

Napisz program, który zgromadzi w kontenerze vector obiekty typu **Punkt**.

Znajdź punkt najbardziej wysunięty w kierunku osi y tj. o największej wartości składowej y .

ZADANIE 6A

Stwórz strukturę Student przechowującą imię i nazwisko (jako string) oraz wektor ocen (typu float). Zaimplementuj funkcję umożliwiającą dodawanie oceny i wypisywania średniej ocen.

ZADANIE 6B (wersja dla ambitnych)

Stwórz strukturę **Student** przechowującą imię i nazwisko (jako string) oraz wektor przechowujący obiekty typu **Ocena**.

Struktura **Ocena** przechowuje:

- nazwę przedmiotu (string)
- ilość punktów ECTS (int)
- ocenę (float).

Zaimplementuj strukturę **Baza** przechowującą wiele obiektów typu **Student**. Zaproponuj funkcje (metody składowe) obsługi takiej bazy:

- Dodawanie studenta,
- Wyszukiwanie studenta,
- Dodawanie oceny studentowi
- Sortowanie studentów (po nazwisku, po ocenie)
- Wypisywanie średniej z danego przedmiotu dla wszystkich studentów
- Wypisywanie średniej ze wszystkich przedmiotów dla danego studenta

Dla bardzo ambitnych: zaimplementuj bazę w oparciu o kontener map, w którym indeks będzie numerem PESEL studenta.








AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski

Lektora do poduszki

- Opis klasy vector z przykładami 
<http://pl.wikibooks.org/wiki/C++/Vector>
- Opis klasy string z przykładami 
<http://pl.wikibooks.org/wiki/C++/String>
- Iteratory z przykładami 
<http://pl.wikibooks.org/wiki/C++/Iteratory>
- Funkcje składowe klasy string z przykładami 
<http://www.cplusplus.com/reference/string/string/>
- Funkcje składowe klasy vector z przykładami 
<http://www.cplusplus.com/reference/vector/vector/>

String

INICJALIZACJA

```
#include <string>
using namespace std;
[...]
string napis;
napis = "text";
```

OPERATORY

Kopiowanie

`a = b`

Porównanie

`a == b`

Dodawanie do końca

`a += b`

empty()	Zwraca wartość true jeżeli napis jest pusty.
size(), length()	Zwraca ilość znaków w napisie.
at()	Zwraca znak o podanym położeniu, tak jak operator [], z tym że ta metoda jest bezpieczniejsza - wyrzuca wyjątek w przypadku wyjścia poza zakres stringa.
clear()	Usuwa wszystkie znaki z napisu.
erase(index, dlugosc)	Usuwa wybrane znaki.
find(podciag)	Znajduje podciąg w ciągu. Funkcja zwraca pozycje, na którym znaleziono podciąg lub std::string::npos.
swap(a,b)	Zamienia miejscami dwa stringi, a staje się b, a b staje się a.
substr(indeks, dlugosc)	Zwraca podciąg na podstawie indeksu początkowego i długości podciągu.
append(napis)	Dodaje zadany napis na końcu istniejącego ciągu.
c_str()	Zwraca napis w stylu języka C (stały wskaźnik typu const char*).

ZADANIE 1

Napisz funkcję która zweryfikuje, czy podany numer dowodu osobistego (jako String) jest prawidłowy.

Numery dowodu osobistego składają się z trzech liter oraz 6 cyfr.

```
char znak = '1';  
if(isdigit(znak)){  
    cout<<"Jest liczba";  
} else {  
    cout<<"Nie jest liczba";  
}
```

```
char znak = 'a';  
if(isalpha(znak)){  
    cout<<"Jest litera";  
} else {  
    cout<<"Nie jest litera";  
}
```

ZADANIE 2

Napisz funkcję która sprawdzi czy w dowolnym zdaniu występują oba podane wyrazy

ZADANIE 3

Napisz funkcję, która odwróci dowolny napis (string).

Np.

"DOWOLNY" -> "YNLOWOD"



std::vector<T> - podstawy

```
#include <vector>
using namespace std;
[...]

//vector<T> nazwaTablicy;
vector<int> vec;
vec.push_back(1);
vec.push_back(5);
vec.push_back(10);
vec.push_back(-2);
vec.push_back(4);

for(int i = 0; i<vec.size(); i++){
    cout<<vec[i]<<endl;
}
```

```
struct Object{
    int x;
};

int main(){
    vector<Object> wObj;
    for(int i = 0; i<4; i++){
        Object objDoWstawienia = {i};
        wObj.push_back(objDoWstawienia);
    }
}
```



`std::vector<T>` - Iteratory

```
vector<int> tab;
```

// inicjujemy wektor kolejnymi liczbami naturalnymi

```
tab.push_back(1);
```

```
tab.push_back(2);
```

```
tab.push_back(3);
```

*// **vector.begin()** – zwraca wskaźnik/iterator do pierwszego elementu wektora*

*// **vector.end()** – zwraca wskaźnik/iterator do elementu będącego **ZA** ostatnim elementem w wektorze*

*// **vector.back()** – zwraca referencje do ostatniego elementu w wektorze*

```
vector<int>::iterator i;
```

```
vector<int>::iterator start = tab.begin();
```

```
vector<int>::iterator koniec = tab.end();
```

```
for( i = start; i != koniec; ++i ) {
```

```
    cout<< *i <<endl;
```

```
}
```

std::vector<T> - Funkcje składowe

void swap(vector<T>& vec)	zamienia zawartości dwóch wektorów miejscami
void push_back(const T obj)	dodaje na końcu wektora kopię przekazanego argumentu
void pop_back()	usuwa ostatni element z wektora
void clear()	usuwa wszystkie elementy z wektora
void assign(size_t n, const T obj)	czyści wektor i wypełnia go n kopiami argumentu obj
void assign(iterator poczatek, iterator koniec)	czyści wektor i wypełnia go elementami z innego wektora z przedziału <poczatek;koniec>
iterator insert(iterator pos, T obj)	wstawia element obj przed wskazywaną przez iterator pos pozycją i zwraca iterator do dostawionego elementu
void insert(iterator pos, size_t n, const T obj)	wstawia n kopii argumentu obj przed pozycją wskazywaną przez iterator pos
void insert(iterator pos, iterator poczatek, iterator koniec)	wstawia przed pozycją wskazywaną przez iterator pos elementy między iteratorami poczatek i koniec (włącznie)
iterator erase(iterator pos)	usuwa element wskazywany przez pos i zwraca iterator do następnego elementu
iterator erase(iterator poczatek, iterator koniec)	usuwa elementy z przedziału <poczatek;koniec> i zwraca iterator do elementu za nimi

std::vector<T> - Przykłady

- Wstawianie wartości na pozycję nr 2

```
vector<int> vec;  
tab.push_back(1);  
tab.push_back(1);  
tab.push_back(1); // 1, 1, 1  
vector<int>::iterator it = vec.begin();  
vec.insert(it+2, 200); // 1, 1, 200, 1
```

- Sortowanie wektora/tablicy

```
#include <algorithm>  
[...]  
int tablica[] = {32,71,12,45,26,80,53,33};  
vector<int> vec(tablica, tablica+8); // kopiowanie tablicy do wektora  
sort(vec.begin(), vec.end()); // 12,26,32,33,45,53,71,80  
// można też sortować część wektora!  
sort(vec.begin(), vec.begin()+4); // 12,32,45,71,26,80,53,33
```

std::vector<T> - Przykłady

- Sortowanie przy użyciu funkcji porównującej (komparatora)

// Funkcja porównująca dwie wartości/obiekty – używana do sortowania
// Można ją wykorzystać do sortowania wektora obiektów ze względu na konkretną zmienną składową!

```
bool comparer (int i, int j) {  
    if(i>j) return true;    // Pierwsza wartość większa od drugiej – gwarantuje sortowanie malejące  
    return false;  
}
```

```
int main(){  
    int tablica[] = {32,71,12,45,26,80,53,33};  
    std::vector<int> vec(tablica, tablica+8); // kopiowanie tablicy do wektora  
    std::sort(vec.begin(), vec.end(), comparer); // 80,71,53,45,33,32,26,12  
}
```


std::vector<T> - Przykłady

- Wyszukiwanie obiektów w wektorze

```
#include <algorithm>
```

```
#include <vector>
```

```
[...]
```

```
vector<int> vec;
```

```
vec.push_back(10);
```

```
vec.push_back(20);
```

```
vec.push_back(15);
```

```
int szukana = 15;
```

```
// find(iterator do początku przedziału, iterator do końca przedziału, szukana wartosc)
```

```
vector<int>::iterator iter = find(vec.begin(), vec.end(), szukana);
```

```
if(iter != vec.end()){
```

```
// różnica między położeniami iteratora początku i znalezionego
```

```
// czyli indeks elementu w wektorze
```

```
int index = distance(vec.begin(), iter);
```

```
cout<<szukana<<" na pozycji "<<index;
```

```
}
```

ZADANIE 4

Stwórz strukturę typu **LiczbaZespolona**.

Utwórz wektor zawierający kilka obiektów typu **LiczbaZespolona**. Posortuj wektor malejąco ze względu na część rzeczywistą liczby zespolonej.

ZADANIE 5

Napisz program, który będzie gromadził w kontenerze vector wpisane z klawiatury imiona. Po wpisaniu słowa "koniec" zostanie wypisane najdłuższe imię.

ZADANIE 6

Napisz program, który zgromadzi w kontenerze vector obiekty typu **Punkt**.

Znajdź punkt najbardziej wysunięty w kierunku osi y tj. o największej wartości składowej y.

ZADANIE 6A

Stwórz strukturę Student przechowującą imię i nazwisko (jako string) oraz wektor ocen (typu float). Zaimplementuj funkcję umożliwiającą dodawanie oceny i wypisywania średniej ocen.

ZADANIE 6B (wersja dla ambitnych)

Stwórz strukturę **Student** przechowującą imię i nazwisko (jako string) oraz wektor przechowujący obiekty typu **Ocena**.

Struktura **Ocena** przechowuje:

- nazwę przedmiotu (string)
- ilość punktów ECTS (int)
- ocenę (float).

Zaimplementuj strukturę **Baza** przechowującą wiele obiektów typu **Student**. Zaproponuj funkcje (metody składowe) obsługi takiej bazy:

- Dodawanie studenta,
- Wyszukiwanie studenta,
- Dodawanie oceny studentowi
- Sortowanie studentów (po nazwisku, po ocenie)
- Wypisywanie średniej z danego przedmiotu dla wszystkich studentów
- Wypisywanie średniej ze wszystkich przedmiotów dla danego studenta

Dla bardzo ambitnych: zaimplementuj bazę w oparciu o kontener map, w którym indeks będzie numerem PESEL studenta.



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Podstawy Informatyki

Informatyka Stosowana – ROK I

mgr inż. Krzysztof Bzowski

Pomiar czasu za pomocą boost::timer

```
#include <boost/timer/timer.hpp>
#include <cmath>
#include <iostream>

using namespace boost::timer;
using namespace std;
int main(){
    cpu_timer t;
    t.start();

    for (long i = 0; i < 10000000; ++i)
        sqrt(123.456L);

    t.stop();
    cout<<"Operacja zajela: "<<t.elapsed().wall / 1e9 <<" s."<<endl;

    return 0;
}
```

Dokumentacja: http://www.boost.org/doc/libs/1_55_0/libs/timer/doc/cpu_timers.html

Biblioteki: <http://sourceforge.net/projects/boost/files/boost-binaries/1.55.0-build2>

Zadanie

Wykorzystując funkcję **sort()** z biblioteki *algorithms* oraz `boost::timer`, sprawdzić skalowalność zaimplementowanego sortowania w bibliotece standardowej dla losowych danych.

Opis losowania liczb znajduje się w LAB 4

Powtórzyć sortowanie 5 razy dla każdej wielkości tablicy i uśrednić wyniki.

Wyniki zapisać w pliku.

Na podstawie pliku stworzyć wykres (np. za pomocą Excella)