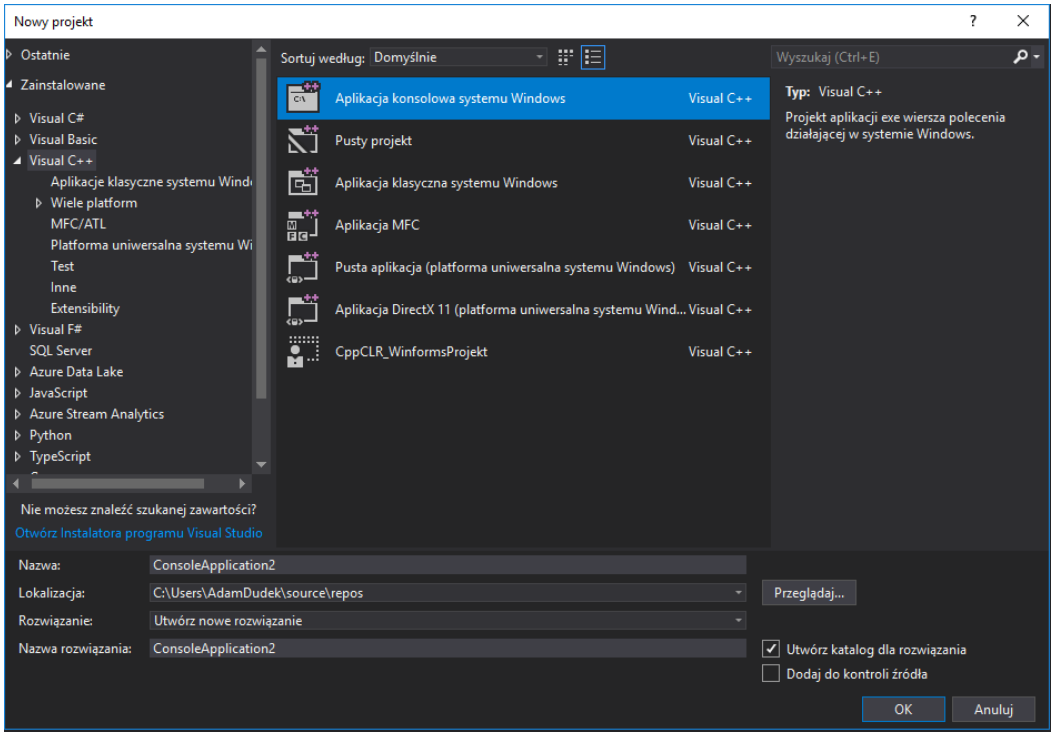
Programowanie II

Lista 2

Przypomnienie semestru poprzedniego i kilka nowości względem klasycznego C

Uruchomienie nowego projektu

Plik->Nowy projekt




```cpp
// ConsoleApplication2.cpp : Ten plik zawiera funkcję „main”. W nim rozpoczyna się i kończy wykonywanie programu.
//

#include "pch.h"
#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}

// Uruchomienie programu: Ctrl + F5 lub menu Debugowanie > Uruchom bez debugowania
// Debugowanie programu: F5 lub menu Debugowanie > Rozpocznij debugowanie

// Porady dotyczące rozpoczynania pracy:
//   1. Użyj okna Eksploratora rozwiązań, aby dodać pliki i zarządzać nimi
//   2. Użyj okna programu Team Explorer, aby nawiązać połączenie z kontrolą źródła
//   3. Użyj okna Dane wyjściowe, aby sprawdzić dane wyjściowe kompilacji i inne komunikaty
//   4. Użyj okna Lista błędów, aby zobaczyć błędy
//   5. Wybierz pozycję Projekt > Dodaj nowy element, aby utworzyć nowe pliki kodu, lub wybierz pozycję Projekt > Dodaj istniejący element, aby dodać istniejące plik
//   6. Aby w przyszłości ponownie otworzyć ten projekt, przejdź do pozycji Plik > Otwórz > Projekt i wybierz plik sln
```

Dodanie ***using namespace std***; pozwala wykorzystać przestrzeń nazw domyślnej obsługi strumienia – wystarczy napisać ***cout*** zamiast ***std::cout;***

UWAGA !!!

W zależności od wersji Visual Studio, w których były przygotowane, przykłady na zajęcia zawierają wpis #include "pch.h" (visual studio 2017 i nowsze) lub #include "stdafx.h" (wersje wcześniejsze).

Stała ale jednak zmienna

```cpp
/*
Stała zmienna ?
W ogólności stałe to wartości nadana na "sztywno" w kodzie programu. W tym przypadku chodzi o zmienne
których wartość nie może zmieniać się w czasie działania programu (możemy nadać im wartość początkową
na etapie uruchamiania programu, ale później nie możemy wprowadzać w nich ŻADNYCH zmian.
*/
#include "stdafx.h"
#include <iostream>;
using namespace std;


int main()
{
    cout << "Podaj wartosc [1-5]" << endl;
    int wartosc;
    cin >> wartosc;
    const int wartosc_stala{ wartosc }; //tworzmy nowy obiekt stałej
    cout <<"Podana wartosc " << wartosc << endl;
    cout << "Podana wartosc po powiekszeniu " << ++wartosc << endl;;
    cout << "Wartosc stala " << wartosc_stala << endl;
    //cout << "Wartosc stala po powiekszeniu " << ++wartosc_stala << endl;
    //wiersz jest w komentarzu, gdyż tak nie wolno nam zrobić
    system("pause");
    return 0;
}
```

```
Podaj wartosc [1-5]
3
Podana wartosc 3
Podana wartosc po powiekszeniu 4
Wartosc stala 3
Press any key to continue . . .
```

Wyrażenie constrexpr

```cpp
/*
Stałe dosłowne - constrexpr
Stałe znane na etapie kompilacji programu - czy nie można użyć po prostu const ?
Można - ale taki program będzie realizowany szybciej - a przecież o to nam chodzi.
*/

#include "stdafx.h"
#include "stdafx.h"
#include <iostream>;
using namespace std;

int main()
{
    constexpr int ilosc_kolumn = 10;
    constexpr int ilosc_wierszy = 20;
    constexpr int ilosc_wszystkich = ilosc_kolumn * ilosc_wierszy;
    cout << "Ilosc elementow " << ilosc_wszystkich << endl;
    system("Pause");
    return 0;
```

```
Ilosc elementow 200
Press any key to continue . . .
```

Typ wyliczeniowy

```cpp
/*
Ty wyliczeniowy - definicja własnego typu pozwalająca rzutować np. liczby całkowite
na jakieś własne identyfikatory
*/

#include "stdafx.h"
#include <iostream>
using namespace std;

int main()
{
    enum class TDzienTygodnia {
        pn = 1,
        wt = 2,
        sr = 3,
        czw = 4,
        pi = 5,
        so = 6,
        nie = 7
    };
    TDzienTygodnia AktualnyDzien;
    AktualnyDzien = TDzienTygodnia::pn;
    switch (AktualnyDzien)
    {
    case TDzienTygodnia::pn:
        {
        cout << "Poniedzialek" << endl;
        break;
        }
    }
    system("Pause");
    return 0;
}
```

```
Poniedzialek
Press any key to continue . . .
```

Automatyczne określanie typów

```cpp
/*
Typy automatycznie określane
Począwszy od standardu C++ 11 kompilatory z nim zgodne muszą radzić sobie
z samodzielnym określaniem typu zmiennej na podstawie typu przypisanej im wartości.
*/

#include "stdafx.h"
#include <iostream>
using namespace  std;


int main()
{
    auto zmienna_calkowita=10;
    auto inna_zmienna_calkowita = 10 * 2;
    auto zmienna_calkowita2 = zmienna_calkowita;
    //auto - typ zostanie okreslony w momencie nadawania wartosci
    double liczba_zmienno_przecinkowa;
    decltype(liczba_zmienno_przecinkowa) liczba_zmienno_przecinkowa2;
    //decltype - pozwala "przechwycic" typ jakiejś innej zmiennej, a potem go wykorzystac
    return 0;
}
```

Rzutowanie typów

```cpp
#include "stdafx.h"
#include <iostream>
using namespace std;


int main()
{
    double liczba_1 = 2.5;
    int liczba_2;
    int liczba_3;
    liczba_2 = liczba_1;
    /*
    konwersja bez rzutowania - pojawia się ostrzeżenie kompilatora o możliwej
    utracie danych
    */
    cout << "Liczba 1: " << liczba_1 << endl;
    cout << "Liczba 2: " << liczba_2 << endl;
    liczba_3 = static_cast<int>(liczba_1);
    /*
    sytuacja podobna jak wyżej, ale tym razem to jest świadome rzutowanie
    kompilator nie ma tutaj żadnych wątpliwości.
    */
    cout << "Liczba 3: " << liczba_3 << endl;
    system("Pause");
    return 0;
}
```

```
Liczba 1: 2.5
Liczba 2: 2
Liczba 3: 2
Press any key to continue . . .
```

Napisy czyli stringi

```
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string napis_1{ "Tresc napis 1" };
    cout << "Napis 1: " << napis_1 << endl;
    string napis_2;
    cout << "Podaj napis 2 (Koniec to koniec)"<<endl;
    getline(cin, napis_2);
    /*
    używamy metody getline() aby można był podać z klawiatury
    napis zawierjący spacje - warto sprawdzić co dzieje się,
    gdy będizemy to realizować przy użyciu cin
    */
    cout << "Napis 2: "<<napis_2 << endl;
    if (napis_2 == "Koniec")
    return 0;
    //porównujemy napisy i przerywamy wykonanie w przypadku gdy wynik jest poprawny
    string napis_3 = napis_1 + " " + napis_2;
    cout <<"Napis 3: "+napis_3 << endl;
    //składamy napisy
    string napis_5, napis_6;
    cout << "Podaj 1 napis do porownania \n";
    cin >> napis_5;
    cout << "Podaj 2 napis do porownania \n";
    cin >> napis_6;
    string wynik_porownania = "Alfabetycznie mniejszy jest ciag ";
```

```
    if (napis_5 <= napis_6)
        wynik_porownania += napis_5;
    else
        wynik_porownania += napis_6;
    cout << wynik_porownania << endl;
    //operator porównania dla napisów
    for (int i = 0; i < 10; i++)
    {
        cout << "To jest powtorzenie nr " + to_string(i + 1) << endl;
    }
    //konwersja liczby na napis
    int dlugosc_napisu = wynik_porownania.size();
    for (int i = 0; i < dlugosc_napisu; i++)
        cout << wynik_porownania[i] << " ";
    cout << endl;
    //rozmiar napisu (ilość znaków)

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

```
Napis 1: Tresc napis 1
Podaj napis 2 (Koniec to koniec)
Tresc napis 2
Napis 2: Tresc napis 2
Napis 3: Tresc napis 1 Tresc napis 2
Podaj 1 napis do porownania
napis1
Podaj 2 napis do porownania
napis2
Alfabetycznie mniejszy jest ciag napis1
To jest powtorzenie nr 1
To jest powtorzenie nr 2
To jest powtorzenie nr 3
To jest powtorzenie nr 4
To jest powtorzenie nr 5
To jest powtorzenie nr 6
To jest powtorzenie nr 7
To jest powtorzenie nr 8
To jest powtorzenie nr 9
To jest powtorzenie nr 10
A l f a b e t y c z n i e   m n i e j s z y   j e s t   c i a g   n a p i s 1
Press any key to continue . . .
```

Klasa string

```cpp
/*
Klasa biblioteczna string jest odpowiedzią na pochodzenie c++ od jezyka c, w którym
napisów jako takich nie było (poza tablicami znaków).
Aby móc korzystać z klasy bibkliotecznej std:string musimy deklarację
tej klasy dołączyć do programu dyrektywą #include <string>.
Warto zwrócić uwagę, że w przeciwieństwie do używanej wcześniej biblioteki
string.h nie uzywamy ".h" - przyjęto, że deklaracje klas bibliotecznych w nowoczesnym C++
umieszczane są w plikach nagłówkowych bez rozszerzenia .h.
*/


#include "pch.h"
#include <iostream>
#include <string>
 using namespace std;

void WyswietlNapis(string koment, string wysw)
{
    cout << "Wyswietlam napis " << koment << ", jego wartosci to: " << wysw << endl;
}
```

```cpp

int main()
{
    string NapisPusty; // utworzenie obiektu klasy napis (bez wartości początkowej)
    WyswietlNapis("NapisPusty", NapisPusty);
    string NapisWstepnieZainicjowany("Wartosc napisu");
    WyswietlNapis("NapisWstepnieZainicjowany", NapisWstepnieZainicjowany);
    char TabPom[10] = { "123456789" };
    /*
    Pomocnicza tablica znaków - 10 elementów ale mieści tylko 9 - ostatni to znak końca
    */
    string NapisZTablicy(TabPom);
    WyswietlNapis("NapisZTablicy", NapisZTablicy);
    string KoncowkaNapisu(&TabPom[3]); // do napisu trafią elementy tablicy powyżej 3
    WyswietlNapis("KoncowkaNapisu", KoncowkaNapisu);
    string PierwszychKilka("To jest jakis tekst", 9);
    WyswietlNapis("PierwszychKilka", PierwszychKilka);
    string AutomatWypel(50, '$');
    WyswietlNapis("AutomatWypel", AutomatWypel);
    string NapisZLiterek({ 'a','b','c' });
    WyswietlNapis("NapisZLiterek", NapisZLiterek);
    WyswietlNapis("Uzycie metody size dla zmiennej NapisZTablicy", to_string(NapisZTablicy.size()));
    //to_string - konwersja liczby na napis, size() zwraca rozmiar wyrażony jako int
    WyswietlNapis("Uzycie metody length dla zmiennej NapisZTablicy", to_string(NapisZTablicy.length()));
    //to_string może otrzymywać parametry różnych typów wbudowanych
    int CalkowitaZNapisu = stoi("12345");
    cout << CalkowitaZNapisu << endl;
    float FloatZNapisu = stof("12.50");
    cout << FloatZNapisu << endl;
```

```cpp
    //konwertujemy napisy na liczby - całkowite i zmienno przecinkowe (stof... - ma wiele różnych końcówek dla różnych typów
    string Wycinek = NapisZTablicy.substr(6, 8);
    WyswietlNapis("Wycinek", Wycinek);
    string NapisPrzeszukiwany("Jakis tekst o jakiejs tresci tekstu");
    int PierwszeWystapieniePodciagu = NapisPrzeszukiwany.find("tekst");
    cout << PierwszeWystapieniePodciagu << endl;
    int OstatnieWystapienie = NapisPrzeszukiwany.rfind("tekst");
    cout << OstatnieWystapienie << endl;
    string Wycinany("Bedziemy sobie wycinac");
    cout << "Napis oryginalny: " << Wycinany << endl;
    cout << "Napis po wycieciu: " << Wycinany.erase(3,5)<< endl;
    Wycinany.pop_back();
    cout << "Napis po usunieciu ostatniego znaku: " << Wycinany<< endl;
    cout << "Napis do dostawieniu czegos po 4 znaku:" << Wycinany.insert(3, "#abcdefgh#") << endl;
    string Literki("abcdefgh");
    string Cyferki("0123456789");
    Literki.replace(3, 2, Cyferki, 4, 5);
    cout << "Efekt zamiany czesci literk na cyferki: " << Literki << endl;
    string Tekst1("Tresc tesktu 1");
    string Tekst2("Tresc tekstu drugiego");
    cout << "Przed zamiana: Tekst1 - " << Tekst1 << ", Tekst 2 - " << Tekst2 << endl;
    swap(Tekst1, Tekst2);
    cout << "Po zamianie: Tekst1 - " << Tekst1 << ", Tekst 2 - " << Tekst2 << endl;
    system("pause");
```

```
Wyswietlam napis NapisPusty, jego wartosci to:
Wyswietlam napis NapisWstepnieZainicjowany, jego wartosci to: Wartosc napisu
Wyswietlam napis NapisZTablicy, jego wartosci to: 123456789
Wyswietlam napis KoncowkaNapisu, jego wartosci to: 456789
Wyswietlam napis PierwszychKilka, jego wartosci to: To jest j
Wyswietlam napis AutomatWypel, jego wartosci to: $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Wyswietlam napis NapisZLiterek, jego wartosci to: abc
Wyswietlam napis Uzycie metody size dla zmiennej NapisZTablicy, jego wartosci to: 9
Wyswietlam napis Uzycie metody length dla zmiennej NapisZTablicy, jego wartosci to: 9
12345
12.5
Wyswietlam napis Wycinek, jego wartosci to: 789
6
29
Napis oryginalny: Bedziemy sobie wycinac
Napis po wycieciu: Bed sobie wycinac
Napis po usunieciu ostatniego znaku: Bed sobie wycina
Napis do dostawieniu czegos po 4 znaku:Bed#abcdefgh# sobie wycina
Efekt zamiany czesci literk na cyferki: abc45678fgh
Przed zamiana: Tekst1 - Tresc tesktu 1, Tekst 2 - Tresc tekstu drugiego
Po zamianie: Tekst1 - Tresc tekstu drugiego, Tekst 2 - Tresc tesktu 1
```

W realizacji zadań proszę wykorzystać przykłady z zajęć. Oczywiście nie zapominamy o podziale programu na odpowiednie funkcje.

Zadanie 1 (6 pkt.)
Napisz program obliczający i drukujący na ekranie objętości i pola powierzchni następujących brył:
a) kuli
b) prostopadłościanu
c) stożka

(Pamiętaj, że użytkownik decyduje o wyborze bryły, a o niezbędne wymiary (i tylko niezbędne) program pyta później)

Zadanie 2 (6 pkt.)
Napisz program, który pyta o oceny z kilku różnych przedmiotów danego studenta, a następnie liczy średnią tych ocen. Sprawdź ponadto, czy studentowi o takiej średniej przysługuje stypendium naukowe (jeśli średnia > 4.1). Należy zapytać użytkownika ile ocen chce podać.

Zadanie 3 (8 pkt.)
Napisz program, który drukuje piramidkę postaci:

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

Głębokość piramidki podaje użytkownik

Zadanie 4 (10 pkt.)
Napisz program, który umożliwi podanie ciągu znaków - jakiegoś zdania a następnie:

- policzy wszystkich ilość liter, ilość liter z pominięciem spacji, ilość liter z pominięciem podanego przez użytkownika znaku
- wypisze wszystkie wyrazy w zdaniu na ekranie
- podzieli zdanie w oparciu o podany przez użytkownika znak (np. przecinek), a wynik podziału wpisze do dynamicznie powiększanej tablicy napisów (nie korzystamy tutaj z możliwości nowego C++, tylko w ramach przypomnienia poprzedniego semestru ręcznie alokujemy pamięć na taką konkstrukcję), np:

zdanie początkowe:
Ala ma kota, a kot ma Alę.
podział w oparciu o spację:

Ala
ma
kota,
a
kot
ma
Alę.

podział w oparciu o przecinek:

Ala ma kota,
a kot ma Alę.