

# Środowisko pracy

---

Pierwszym zadaniem jest przygotowanie środowiska pracy na najbliższe dwa semestry.

Do prac będziemy wykorzystywać:

- zintegrowane środowisko programowania (IDE – *Integrated Development Environment*) Code::Blocks (dla jednolitości komunikacji: w wersji 20.03 32-bitowej).
- zestaw narzędzi MinGW (kompilatory, konsolidatory, debugger).

Naturalne wydaje się przeprowadzenie regularnej instalacji całości potrzebnego nam oprogramowania, ale nie w każdej sytuacji możemy sobie na to pozwolić (np. w laboratorium nie mamy do tego wystarczających uprawnień). Wykorzystamy zatem wersję, która nie wymaga wykonania regularnej instalacji (ani żadnych dodatkowych uprawnień).

**Podaję przepis dla Windows – osoby korzystające z innych systemów operacyjnych, proszę o kontakt.**

## Pobranie Code::Blocks i MinGW

---

Zarówno Code::Blocks, jak i MinGW można pobrać ze strony:

<http://www.codeblocks.org/downloads/binaries>

Trzeba przyjrzeć się dokładnie nazwom pakietów. Ten, którego potrzebujemy, to:

`codeblocks-20.03mingw-32bit-nosetup.zip`

Nazwa tego archiwum zip wyjaśnia chyba wszystko: mamy w nim CodeBlocks (w wersji 20.03), pakiet programów MinGW w wersji 32-bitowej, a całość zamiast instalować wystarczy rozpakować.

Archiwum możemy pobrać z serwerów FossHUB lub Sourceforge.net (nie ma znaczenia, z którego weźmiemy binaria).

Archiwum małe nie jest (183 MB) i proces jego pobierania trochę potrwa (ja czekałem 5 minut).

W jakim folderze (katalogu) jest zapisane archiwum?

# Rozpakowanie środowiska pracy

---

Bardzo istotną sprawą jest zaplanowanie sensownej lokalizacji na katalog, w którym znajdzie się środowisko pracy (wszystkie jego składniki). W przyszłości będziemy dostawać się do tego katalogu z linii poleceń i byłoby nieźle, gdyby spełnić dwa warunki:

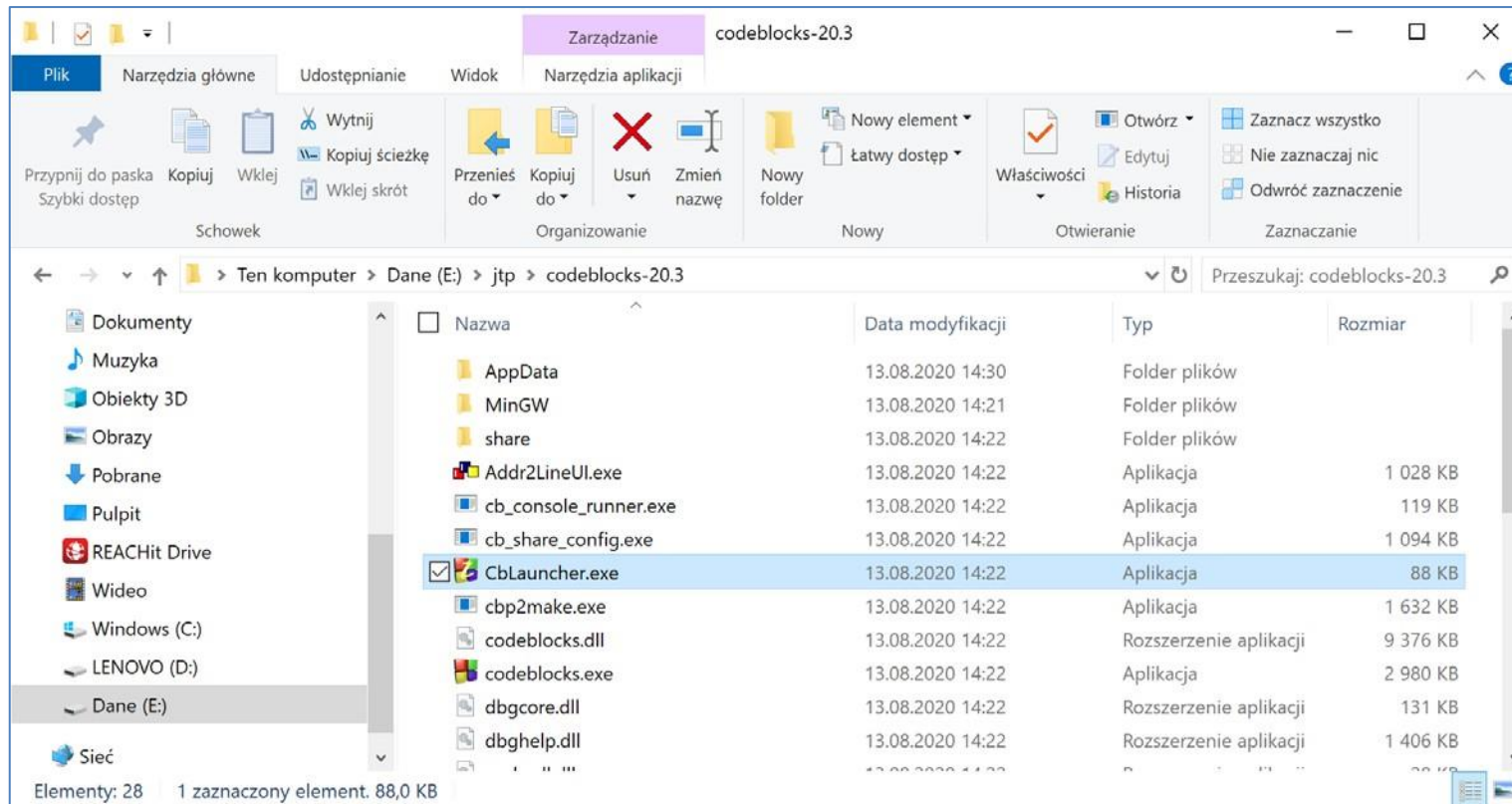
- brak spacji i polskich znaków w ścieżce dostępu do programów środowiska,
- stosunkowo krótka ścieżka dostępu.

Ale najważniejsze, żeby wiedzieć, w którym katalogu jest środowisko i tworzone pod nim programy.

Ja utworzę katalog **jtp** (bezpośrednio w głównym katalogu \ na dysku C: mam dość nieduży), w którym utworzę katalog **codeblocks-20.03** i do niego rozpakuję archiwum (rozpakowanie trwało prawie 10 minut, bo środowisko zajmuje na dysku prawie 600 MB).

# Środowisko – wykończenie

Po zajrzeniu do foldera widzę:



Wysłałam skrót CbLauncher.exe na Pulpit (w menu kontekstowym: Wyślij do > Pulpit (utwórz skrót)).

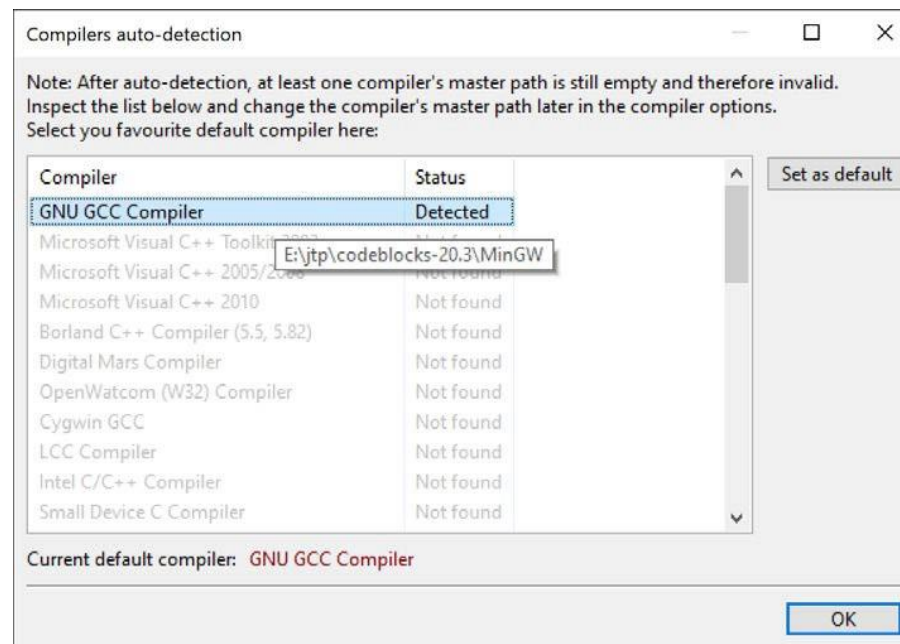
# Uruchomienie

Zmieniam jeszcze podpis skrótu na pulpicie i mam wygodny sposób



uruchomienia C::B:

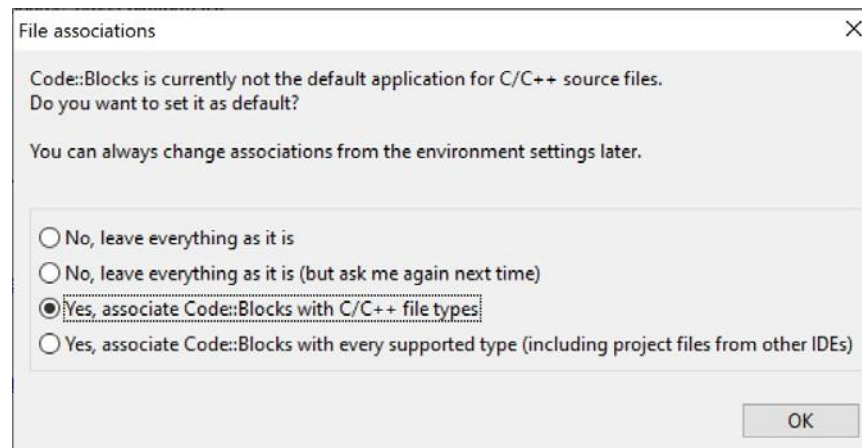
Zanim C::B faktycznie wystartuje, trzeba jeszcze wskazać domyślny kompilator (Set as default i OK).



# C::B jeszcze konfiguracja

---

Po uruchomieniu C::B domaga się jeszcze jednej informacji dotyczącej powiązania typów plików (w Windows są rozpoznawane po rozszerzeniu nazwy pliku) z C::B. Najsensowniejsze wydaje się:



Oznacza ono, że po dwukrotnym kliknięciu na pliki z rozszerzeniami .h, .hpp, .c, .cpp (także .cbp – projekty C::B) będzie uruchomiony C::B.

Teraz będzie można wreszcie napisać pierwszy program.

# Lokalizacja plików

---

Katalog jtp utworzyłem z myślą o tym, żeby mieć tam wszystkie elementy związane z przedmiotem: i oprogramowanie, i kody źródłowe.

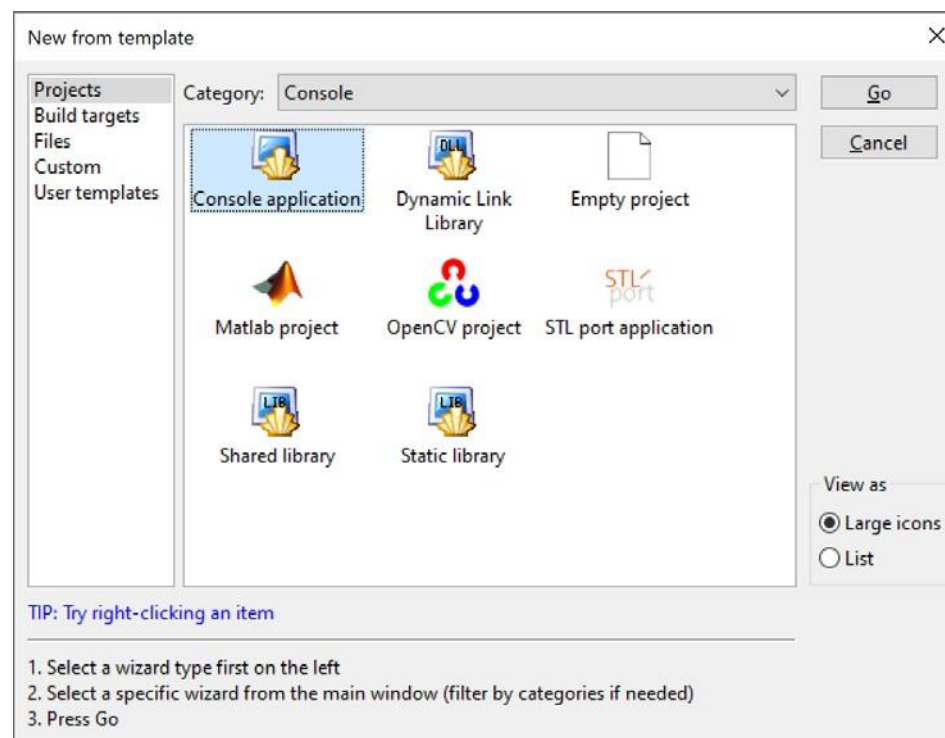
Tworzę w jtp kolejny katalog project i w nim będę tworzył kolejne projekty w trakcie semestru.

Teraz jestem gotowy do utworzenia w C::B nowego projektu, który nazwę zad\_m1 (bo to pierwsze z małych zadań).

Poleceniem File > New > Project uruchamiam kreator nowego projektu i od razu staję przed pytaniem, który z 40 (jeśli nie pomyliłem się w liczeniu) rodzajów projektu wybrać. Warto poświęcić chwilę czasu na przejrzenie tej listy i pokombinować z kategoriami projektów.

# Aplikacja konsolowa

To, co nas teraz interesuje, to:



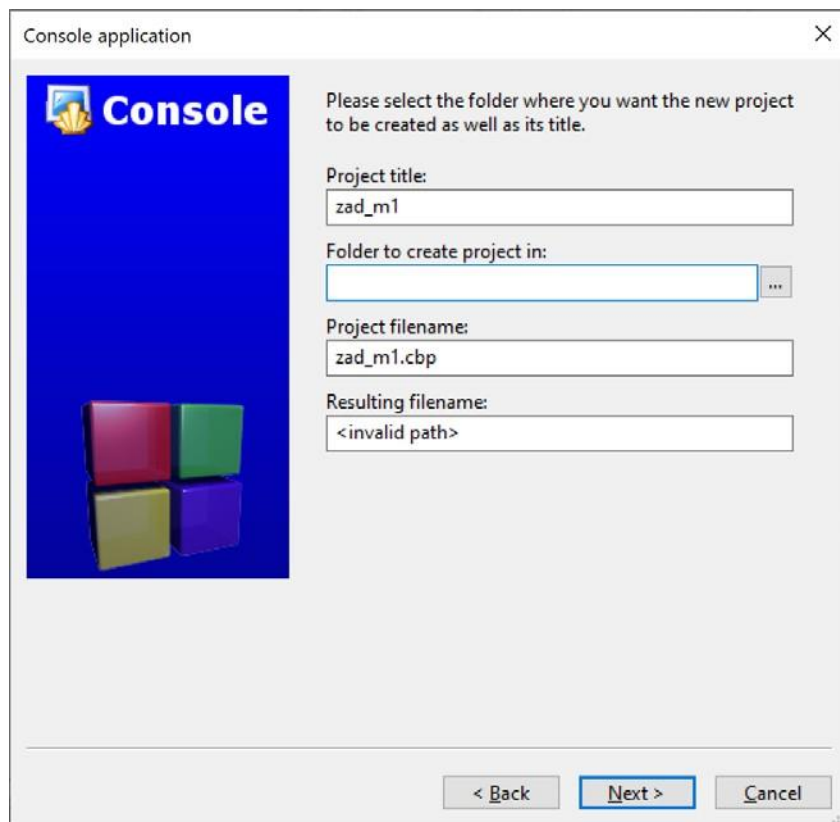
Po wybraniu Console application możemy przejść do kolejnych kroków naciskając przycisk **Go**.



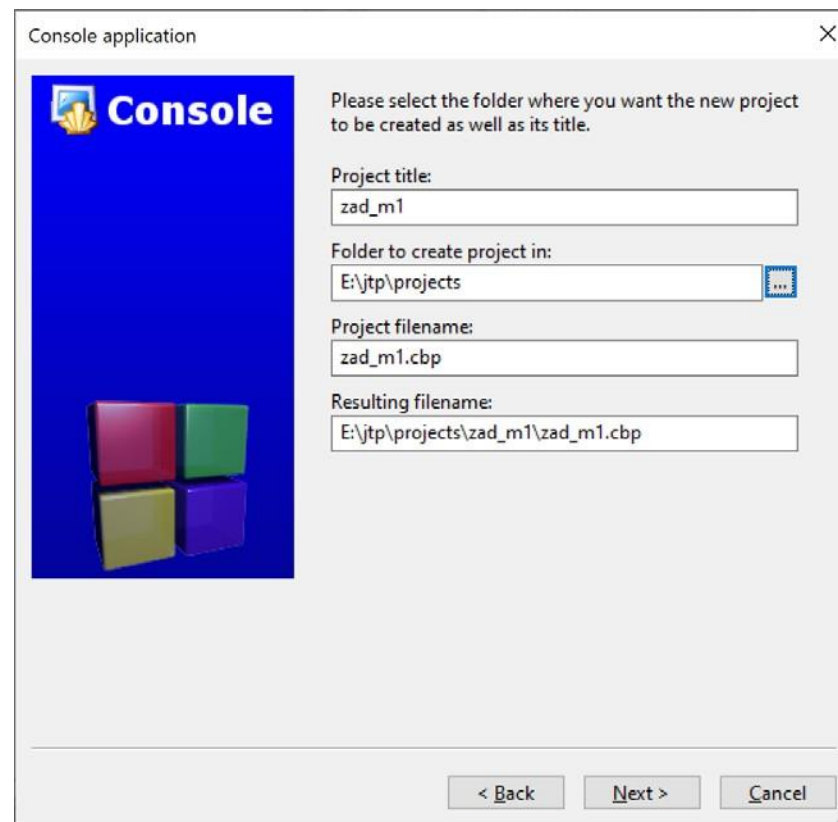
# Język C++ i nazwa projektu.

Normalnie wyłączam od razu wyświetlanie okienka powitalnego, ale odłożę to tymczasem. Przyciskiem **Next** przechodzę na kolejną stronę, na której język C++ jest wybrany domyślnie.

Na kolejnej stronie trzeba nam wybrać lokalizację i nazwę projektu:



The screenshot shows the 'Console application' wizard window. The title bar says 'Console application'. The main area has a blue background with the 'Console' logo and a Windows logo. The text says: 'Please select the folder where you want the new project to be created as well as its title.' The fields are: 'Project title:' with 'zad\_m1', 'Folder to create project in:' with an empty text box and a browse button (...), 'Project filename:' with 'zad\_m1.cbp', and 'Resulting filename:' with '<invalid path>'. At the bottom are buttons: '< Back', 'Next >', and 'Cancel'.



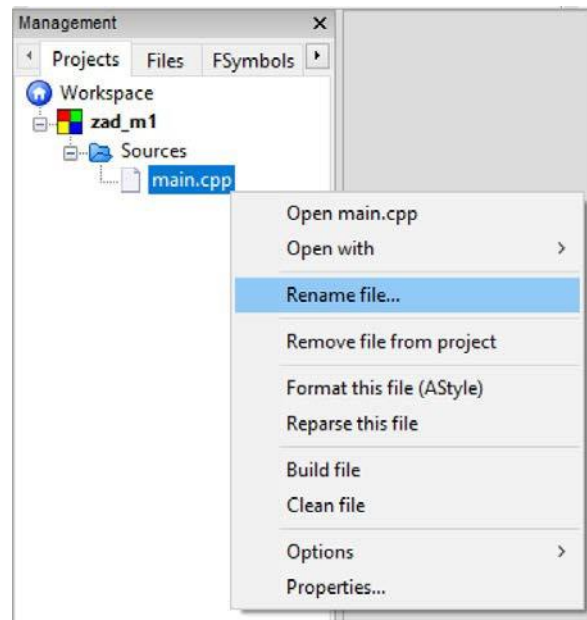
The screenshot shows the 'Console application' wizard window, Step 2. The title bar says 'Console application'. The main area has a blue background with the 'Console' logo and a Windows logo. The text says: 'Please select the folder where you want the new project to be created as well as its title.' The fields are: 'Project title:' with 'zad\_m1', 'Folder to create project in:' with 'E:\jtp\projects' and a browse button (...), 'Project filename:' with 'zad\_m1.cbp', and 'Resulting filename:' with 'E:\jtp\projects\zad\_m1\zad\_m1.cbp'. At the bottom are buttons: '< Back', 'Next >', and 'Cancel'.

# Zmiana nazwy pliku programu

---

Na ostatniej stronie kreatora, w ustawieniach konfiguracji, nic nie zmieniamy i przyciskiem **Finish** faktycznie tworzymy projekt. Warto poświęcić chwilę na rozejrzenie się w interfejsie i wykrycie zmian.

Chciałbym, żeby główne pliki projektów miały taką samą nazwę, jak projekt, zatem:



# Poprogramowanie

---

Po zmianie nazwy pliku na `zad_m1.cpp` mogę przejść do programowania.

Jedyną rzeczą do zrobienia jest zmiana wyświetlanego tekstu na:  
`imię nazwisko (nr albumu)`

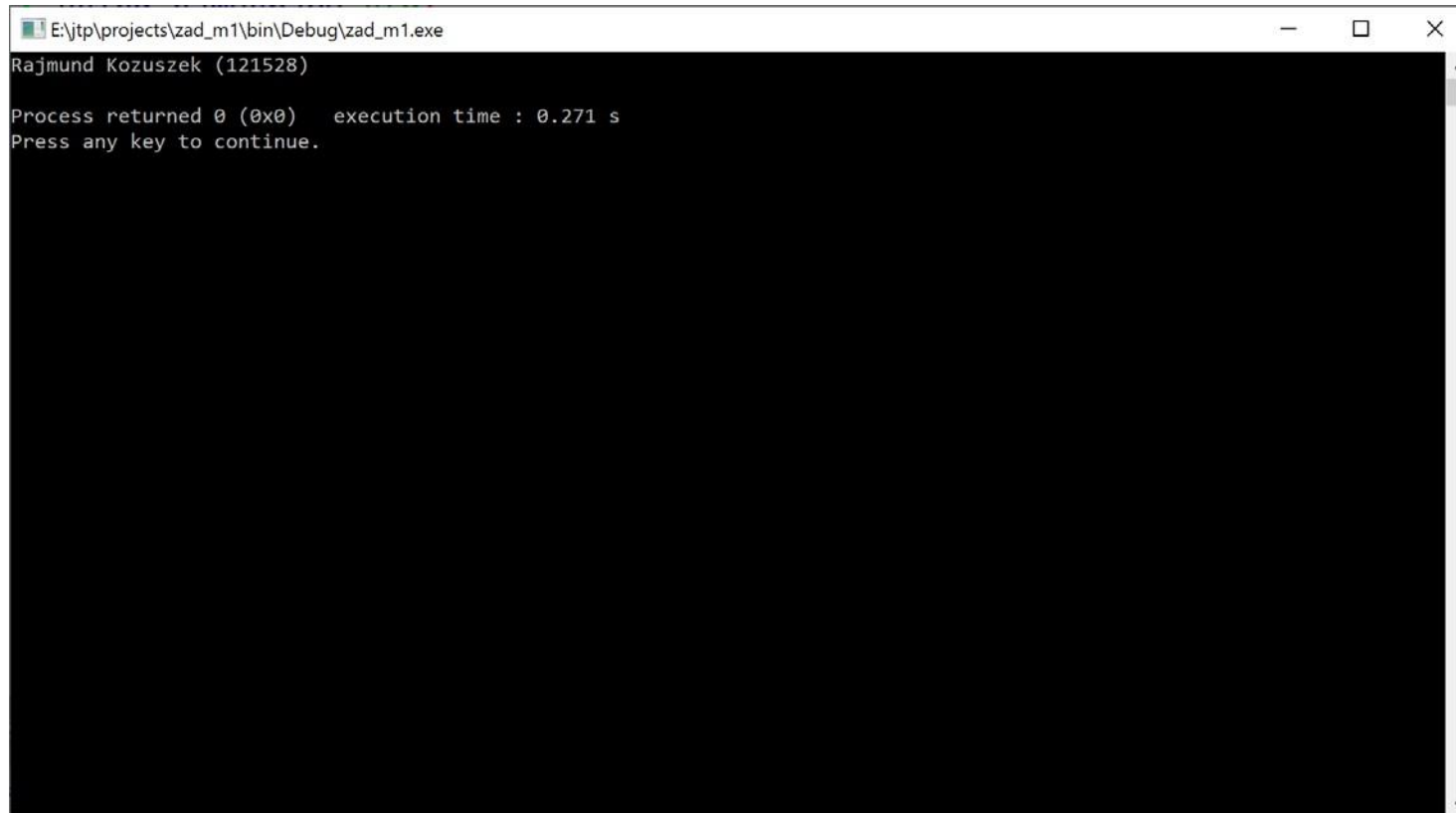
Ta zmiana jest tak oczywista, że nie ma się co nad nią rozwodzić. Jeśli chcecie Państwo uniknąć problemów z kodowaniem, to zastąpcie polskie diakrytyki ich angielskimi odpowiednikami (ja tak zrobiłem).

Proszę poświęcić chwilę na rozejrzenie się w poleceniach związanych z kompilacją i uruchomieniem programu – mamy też przyciski na paskach narzędzi i skróty klawiaturowe do skojarzenia z tymi poleceniami.

# Wynik działania

---

Wynik działania mojej wersji programu jest następujący:



The screenshot shows a Windows command prompt window with the title bar "E:\jtp\projects\zad\_m1\bin\Debug\zad\_m1.exe". The window contains the following text:

```
Rajmund Kozuszek (121528)  
  
Process returned 0 (0x0)   execution time : 0.271 s  
Press any key to continue.
```

Zostaje rzucić efekt pracy do pliku w formacie png.

# Zrzut ekranu

---

Zebranie zrzutu ekranu jest stosunkowo proste, ale dla porządku podaję tutaj kolejne kroki:

1. Po uruchomieniu programu okienko konsoli powinno być aktywne, ale na wszelki wypadek można kliknąć w pasek tytułu tego okna.
2. Skrótami Alt-PrntScr (jak to jest pod Linuxem?) przenosi się obraz aktywnego okna do schowka Windows.
3. Teraz uruchamiam program Paint i ustawiam wielkość obrazu na dużo mniejszą, niż okienko konsoli.
4. Wklejam zawartość schowka skrótami Ctrl-V.
5. Zapisuję plik jako Obraz PNG w dobrze znanej lokalizacji (domyślnej czy nie, nie ma znaczenia – powinniście po prostu wiedzieć, skąd wziąć załącznik do maila).

## Oddawanie zadania

---

Uzyskanie 2 punktów z tego zadania wymaga załadowania w Moodle (zadanie Zrzut ekranu) zrzutu (ang. dump) okienka konsoli z wynikiem wykonania programu. Obraz powinien być w formacie png, a nazwa pliku to numer albumu (indeksu) autora.

Zrzut okna konsoli należy załadować do:

**12 października 2022 23:59**