Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Politechnika Świętokrzyska
Wydział Zarządzania i Modelowania Komputerowego

Kierunek studiów:
Inżynieria Danych

Sławomir Koczubiej

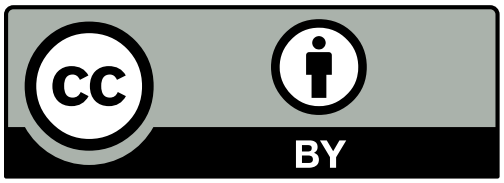
Materiały dydaktyczne do przedmiotu

PODSTAWY PROGRAMOWANIA

opracowane w ramach realizacji Projektu
„Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
FERS.01.05-IP.08-0234/23

Kielce, 2025

Spis treści



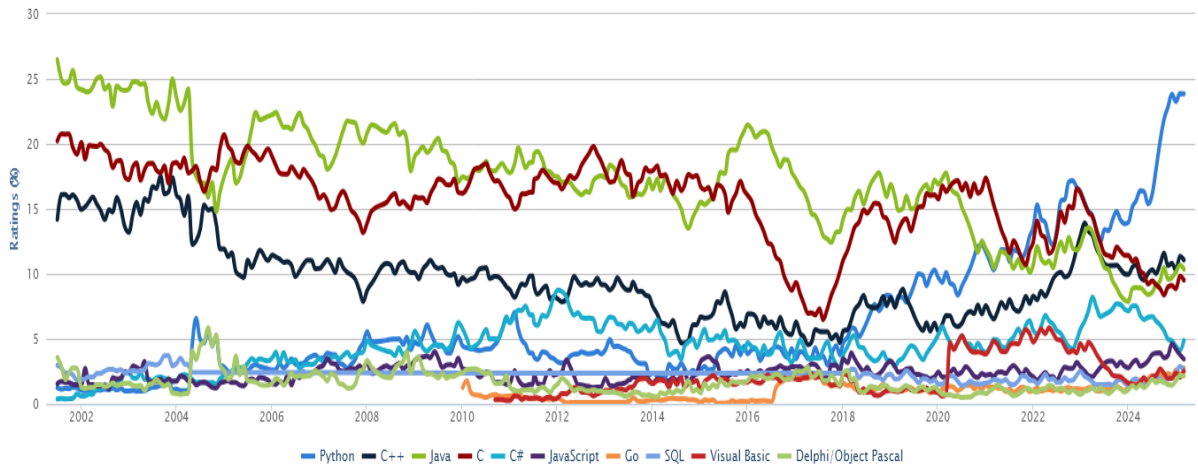
1. Uwagi wstępne

Materiały adresowane są przede wszystkim do studentów uczących się języka programowania C w ramach przedmiotu podstawy programowania. Ponadto mogą być pomocne dla programistów innych języków chcących szybko nauczyć się podstaw C. Ponieważ niniejszy zbiór jest przeznaczony przede wszystkim dla początkujących programistów, wiele spośród zawartych w nim zadań ma służyć nie tyle sprawdzeniu znajomości języka C, co wyrobieniu umiejętności algorytmicznego myślenia i programowania w ogóle. Z tego powodu w zbiorze jest niewiele zadań sprawdzających znajomość funkcji bibliotecznych.

2. Język C

Język C został stworzony w 1972 roku przez Dennisa Ritchiego z firmy Bell Labs w trakcie prowadzonych wspólnie z Kenem Thompsonem prac nad systemem operacyjnym UNIX (Prata, 2016). Ritchie nie wymyślił jednak C zupełnie od podstaw - oparł go na języku B stworzonym przez Thompsona. Ważną rzeczą jest fakt, iż C został pomyślany jako narzędzie pracy programistów, a więc jego podstawowym celem jest użyteczność. W ciągu pięciu już dekad język C stał się jednym z najważniejszych i najpopularniejszych narzędzi programowania (rys. 1) według indeksu TIOBE (https://www.tiobe.com/tiobe-index).

[Tekst alternatywny. Wykres dwuwymiarowy z kilkoma seriami. Opis osi x: kolejne lata od 2002 do 2025; opis osi y: popularność w procentach poszczególnych języków programowania, takich jak: Python, C++, Java, C#, JavaScript, Go, SQL, Visual Basic, Delphi/Object Pascal.]



Rys. 1. Popularność języków programowania

C jest nowoczesnym językiem udostępniającym wszystkie funkcje sterujące, które uznawane są za pożądane w teorii i praktyce programowania. Cechy języka C sprawiają, iż stosowanie takich technik, jak programowanie zstępujące, programowanie strukturalne czy projektowanie modularne, staje się bardziej naturalne. Pozwala to tworzyć bardziej niezawodne i zrozumiałe programy.

C jest językiem efektywnym, potrafiącym wykorzystać możliwości tkwiące we współczesnych komputerach. Programy w języku C odznaczają się zwykle niewielką objętością i dużą szybkością działania. Język C charakteryzuje bowiem wysoki stopień precyzji, utożsamianej zwykle z asemblerem (mnemonicznym zapisem wewnętrznego zestawu instrukcji używanego przez dany procesor).

C jest językiem przenośnym. Oznacza to, że programy napisane w C na jednej platformie systemowej mogą być skompilowane i uruchamiane na innych platformach natychmiast lub po minimalnych modyfikacjach. W przypadku, jeśli modyfikacje są konieczne, często sprowadzają się one do wprowadzenia kilku zmian w plikach nagłówkowych (*header file*) dołączonym do programu głównego. Kompilatory C są dostępne dla wielu platform systemowych - od 8-bitowych mikroprocesorów do superkomputerów. Należy tu jednak zastrzec, że fragmenty programów realizujące bezpośredni dostęp do urządzeń sprzętowych lub wykorzystujące specyficzne funkcje systemu operacyjnego zwykle nie nadają się do przenoszenia pomiędzy platformami.

Równocześnie C pozwala popełniać błędy, które w innym języku programowania byłyby niemożliwe. Język C daje więc programiście więcej swobody, ale i obarcza go większą odpowiedzialnością. Większość implementacji języka C zawiera także duży zbiór przydatnych funkcji, pozwalających w szybki sposób uporać się z wieloma typowymi zadaniami.

3. Środowisko pracy

Zwykle podczas pracy programiści używają (zintegrowanych) środowisk programistycznych (IDE, *integrated development environment*). To program lub grupa programów służących do tworzenia, modyfikowania, testowania i konserwacji oprogramowania. Środowiska programistyczne charakteryzują się tym, że udostępniają złożoną, wieloraką funkcjonalność obejmującą edycję kodu źródłowego, kompilowanie kodu źródłowego, tworzenie zasobów programu (szablonów, okien dialogowych, menu, raportów, elementów graficznych jak ikony lub obrazy), tworzenie baz danych, komponentów i innych.

Środowisko programistyczne zwykle występuje jako osobny pakiet oprogramowania. Przykładami są: Visual Studio, Eclipse, IntelliJ IDEA, Anjuta, Code::Blocks, Kdevelop, Lazarus czy Qt Creator, które obsługują różne języki programowania i wspierają

różne dodatkowe biblioteki programistyczne. Wymienione środowiska programistyczne znacząco ułatwiają pracę w przypadku budowania dużych aplikacji. W przypadku nauki i tworzenia jedno-plikowych programów warto stosować prostsze, nie wymagające dodatkowego nakładu pracy na naukę rozwiązania.

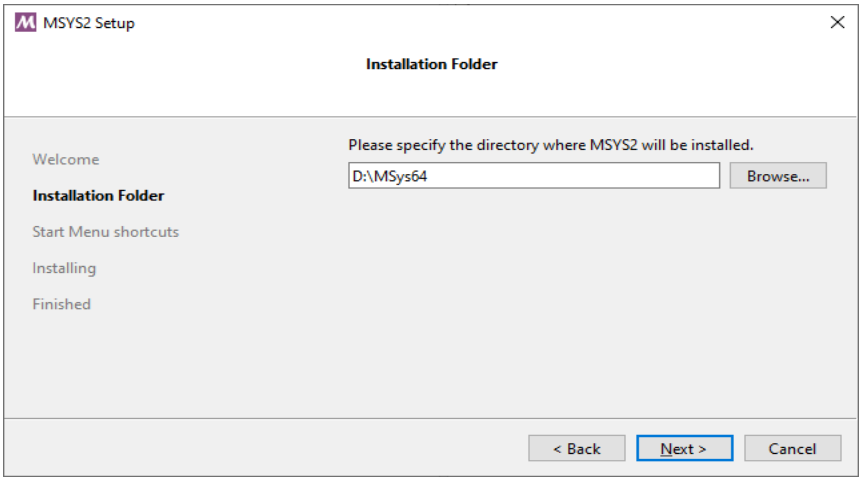
Ciekawym środowiskiem jest **MSYS2** - zbiór narzędzi i bibliotek znanych z systemu GNU/Linux zapewniających łatwe w użyciu środowisko do budowania, instalacji i uruchamiania natywnych aplikacji dla systemu operacyjnego Windows. Środowisko jest darmowe, ma minimalne wymagania sprzętowe. W skład wchodzą: terminal wiersza poleceń **mintty**, **bash**, systemy kontroli wersji takie jak **git** i **subversion**, narzędzia takie jak **sed**, **awk**, **grep** czy system kompilacji **autotools**. MSYS2 zapewnia aktualne natywne kompilacje m.in.: kompilatora **GCC**, **CMake**, **Meson**, **OpenSSL**, **FFmpeg**, wspiera też inne języki programowania jak **Rust**, **Ruby** czy **Python**. Dodatkowym atutem MSYS2 jest fakt, że sposób budowania aplikacji w systemach operacyjnych Windows i GNU/Linux jest taki sam, co ułatwia migrację aplikacji.

Środowisko MSYS2 można łatwo skonfigurować do pracy z popularnym darmowym edytorem programisty **Visual Studio Code** lub jego odmianą **VSCodium**.

4. Instalacja i konfiguracja MSYS2

Instalator można znaleźć pod adresem https://www.msys2.org, będzie miał nazwę **`msys2-x86_64-xxxxxxxx.exe`**, gdzie ciąg znaków **`x`** będzie datą wydania. Po pobraniu instalatora, należy go uruchomić i wskazać miejsce instalacji. Ścieżka instalacji nie powinna zawierać spacji, polskich liter ani innych liter z dodatkowymi znakami diakrytycznymi (rys. 2).

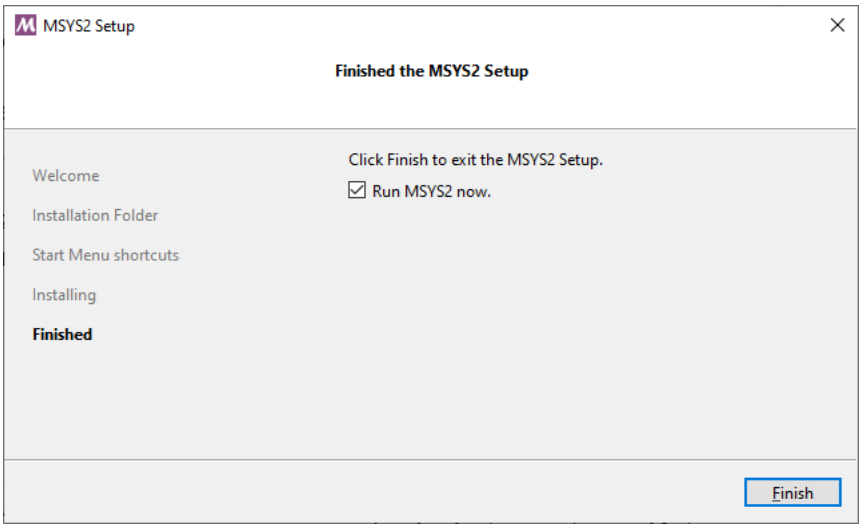
[Tekst alternatywny. Zrzut z ekranu. Okno instalatora, konfiguracja ścieżki instalacji.]



Rys. 2. Konfiguracja ścieżki instalacji

Po zakończeniu procesu instalacji, należy zaznaczyć **Run MSYS2 now.** i kliknąć przycisk **Finish** (rys. 3).

[Tekst alternatywny. Zrzut z ekranu. Okno instalatora, kończenie instalacji.]



Rys. 3. Kończenie instalacji

Następnie uruchomi się terminal poleceń jak na rys. 4, gdzie można między innymi zarządzać pakietami oprogramowania.

[Tekst alternatywny. Zrzut z ekranu. Okno terminala MSYS2.]

Rys. 4. Okno terminala MSYS2

Po uruchomieniu terminala należy zaktualizować lokalne repozytoria pakietów oprogramowania poleceniem **`pacman -Sy`** (jak na listingu 1), następnie

zaktualizować środowisko poleceniem **`pacman -Su`** (listing 2). Kolejnym krokiem jest instalacja kompilatora wraz z dodatkowymi narzędziami (*toolchain*) poleceniem **`pacman -S mingw-w64-ucrt-x86_64-toolchain`**. Ostatnie polecenie spowoduje zainstalowanie zestawu kompilatorów z rodziny **GNU GCC** z biblioteką C **ucrt** (*Microsoft Universal C Runtime*) i debugerem **GNU GDB**. Taki zestaw narzędzi umożliwia kompilowanie kodu dla modelu procesorów z rodziny **x86_64** pod system operacyjny Windows.

```
$ pacman -Sy
:: Synchronizing package databases...
 mingw64             458.1 KiB   237 KiB/s 00:02 [###############] 100%
 ucrt64              498.1 KiB   530 KiB/s 00:01 [###############] 100%
 msys                366.5 KiB   679 KiB/s 00:01 [###############] 100%
...
```

Listing 1. Aktualizacja lokalnych repozytoriów pakietów oprogramowania

```
$ pacman -Su
:: Starting core system upgrade...
warning: terminate other MSYS2 programs before proceeding
resolving dependencies...
looking for conflicting packages...

Packages (3) filesystem-2025.02.23-1  mintty-1~3.7.8-1
             msys2-runtime-3.5.7-4

Total Download Size:   2.75 MiB
Total Installed Size:  8.82 MiB
Net Upgrade Size:      0.00 MiB

:: Proceed with installation? [Y/n]
```

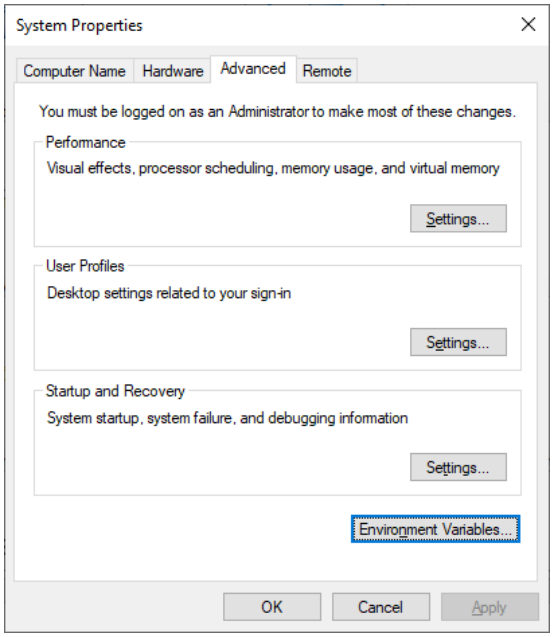
Listing 2. Aktualizacja środowiska MSYS2

```
$ pacman -S mingw-w64-ucrt-x86_64-toolchain
resolving dependencies...
looking for conflicting packages...

Packages (16) mingw-w64-ucrt-x86_64-binutils-2.44-1
              mingw-w64-ucrt-x86_64-gcc-libs-14.2.0-3
...

Total Download Size:    65.86 MiB
Total Installed Size:  519.42 MiB

:: Proceed with installation? [Y/n]
```

Listing 3. Instalacja kompilatora

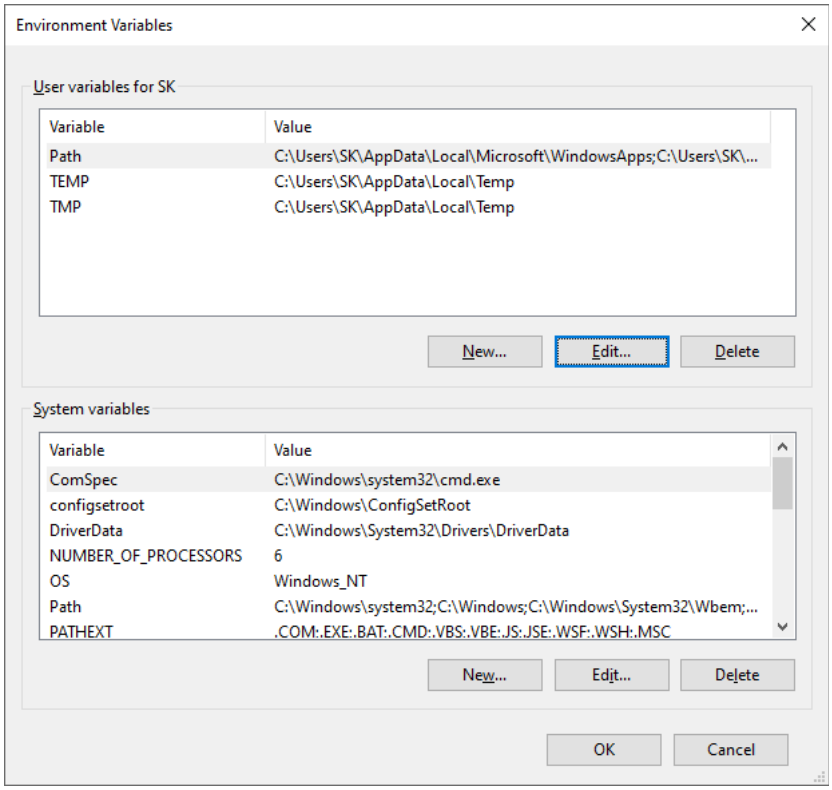
Ostatnim krokiem jest dodanie ścieżki instalacyjnej do zmiennej wyszukiwania **`Path`** systemu operacyjnego. W zaawansowanych ustawieniach systemu (rys. 5) należy

[Tekst alternatywny. Zrzut z ekranu. Okno zaawansowanych ustawień systemu.]



Rys. 5. Okno zaawansowanych ustawień systemu

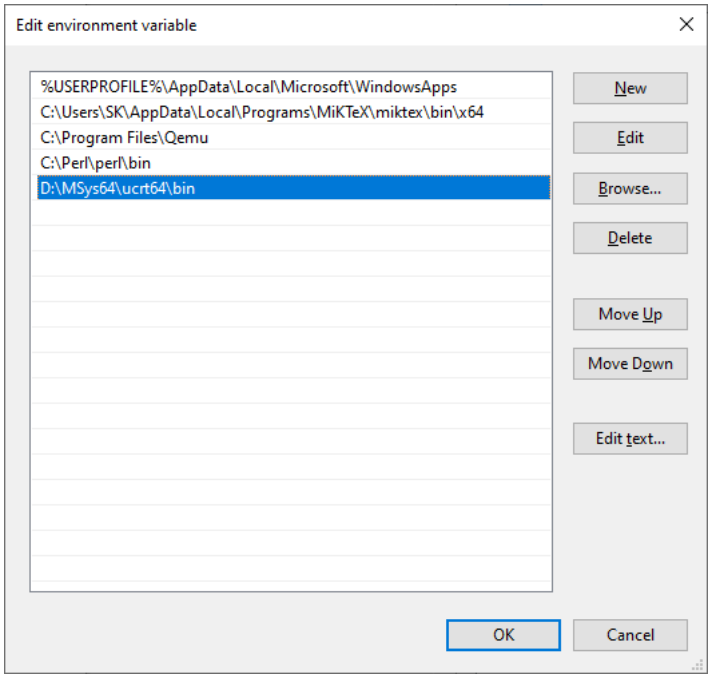
[Tekst alternatywny. Zrzut z ekranu. Okno edycji zmiennych systemowych.]



Rys. 6. Okno edycji zmiennych systemowych

kliknąć zmienne systemowe. Następnie edycja zmiennej **Path** dla użytkownika jak na rysunku 6 i dodanie nowej ścieżki zgodnej z miejscem instalacji MSYS2, jak na rysunku 7.

[Tekst alternatywny. Zrzut z ekranu. Dodanie wartości do zmiennej systemowej Path.]



Rys. 7. Dodanie wartości do zmiennej systemowej **Path**

Więcej informacji dotyczących zarządzania środowiskiem, możliwości konfiguracji kompilatora do różnych systemów operacyjnych i modeli procesora można znaleźć w dokumentacji środowiska (What is MSYS2?).

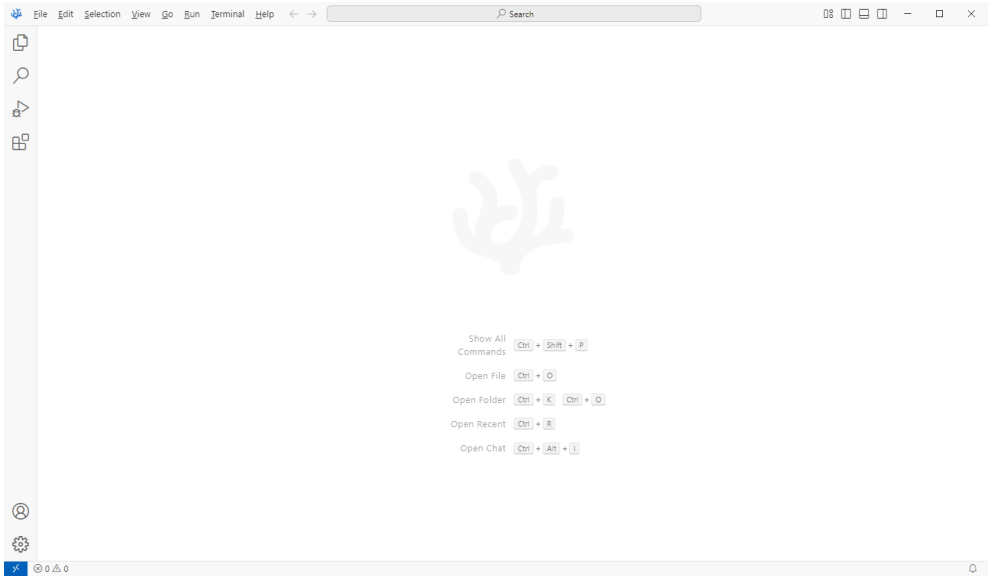
5. Instalacja i konfiguracja VSCodium

VSCodium jest darmowym edytorem programisty, wersją **Visual Studio Code** - bardzo popularnym i cenionym darmowym edytorem programisty z otwartym źródłem firmy Microsoft. Od swojego pierwowzoru różni się tym, że jest pozbawiony modułów telemetrycznych. Instalacja, konfiguracja i sposób użycia jest taki sam w obu przypadkach. Dalej zostanie pokazane jak dostosować edytor VSCodium do pracy ze środowiskiem MSYS2.

Instalator można pobrać ze strony https://github.com/VSCodium/vscodium/releases, w wersjach umożliwiających zainstalowanie programu w systemie operacyjnym, dla pojedynczego użytkownika lub nie wymagającej instalacji (wersja przenośna). Przy pierwszym uruchomieniu edytor wygląda jak na rysunku 8. Edytor można skonfigurować ręcznie do pracy z niemalże każdym dowolnym kompilatorem i debugerem, niemniej jednak łatwiej jest skorzystać z gotowych dodatków. Do pracy

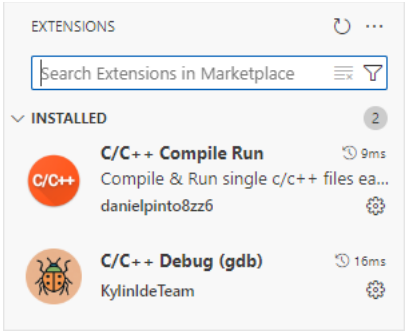
z kompilatorem i debugerem C/C++ ze środowiska MSYS2, w panelu dodatków należy znaleźć i zainstalować dwa rozszerzenia o nazwach: **C/C++ Compile Run** i **C/C++ debug (gdb)**, jak na rys. 9.

[Tekst alternatywny. Zrzut z ekranu. Główne okno edytora programisty.]



Rys. 8. Główne okno edytora programisty VSCodium

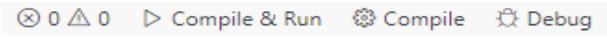
[Tekst alternatywny. Zrzut z ekranu. Okno instalacji dodatków.]



Rys. 9. Zainstalowane dodatki

Po zainstalowaniu dodatków, podczas pisania kodu w dolnej części okna edytora pojawią się dodatkowe przyciski **Compile & Run**, **Compile** oraz **Debug**, jak na rys. 10.

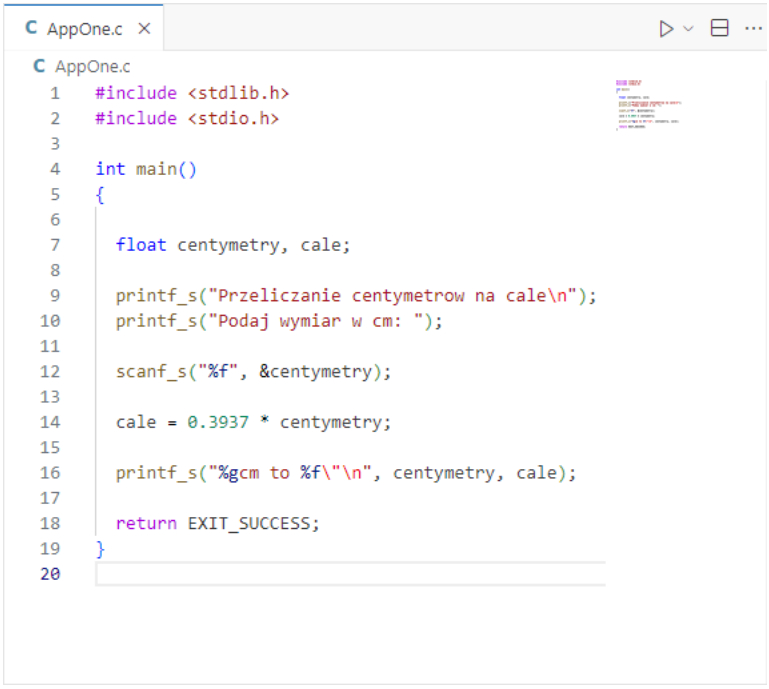
[Tekst alternatywny. Zrzut z ekranu. Okno dodatkowych przycisków do kompilowania, uruchamiania i debugowania programu.]



Rys. 10. Dodatkowe przyciski

Na rysunku 11 pokazano okno z kodem źródłowym prostego programu. Po poprawnym skompilowaniu programu można go uruchomić z poziomu edytora VSCodium.

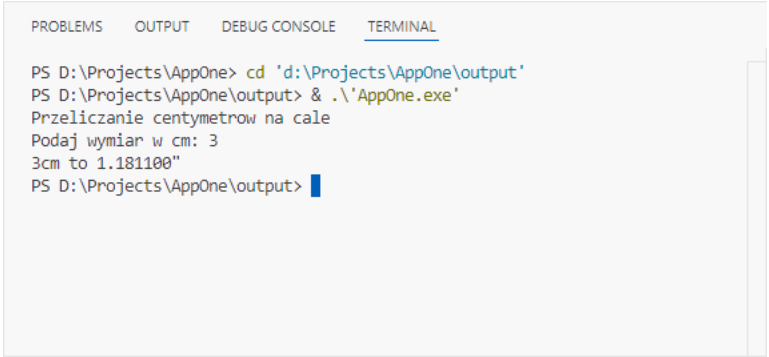
[Tekst alternatywny. Zrzut z ekranu. Okno z kodem źródłowym programu.]



Rys. 11. Okno z kodem źródłowym programu

Komunikacja z programem odbywać się będzie za pomocą wbudowanego w edytor terminala, jak na rysunku 12.

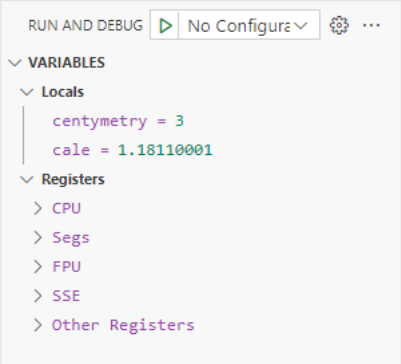
[Tekst alternatywny. Zrzut z ekranu. Panel wbudowanego terminala.]



Rys. 12. Panel wbudowanego terminala.

W szukaniu błędów w programie znacząco pomaga debuger, na rysunku 13 pokazano panel debugera w trakcie śledzenia wartości zmiennych lokalnych programu.

[Tekst alternatywny. Zrzut z ekranu. Panel śledzenia zmiennych programu.]



Rys. 13. Panel śledzenia zmiennych programu

6. Zadania

1. Napisz program, który wczytuje ze standardowego wejścia nieujemną liczbę wymierną x i wypisuje na standardowym wyjściu $\sqrt{x}$.
2. Napisz program, który wczytuje ze standardowego wejścia trzy liczby całkowite i wypisuje na standardowym wyjściu ich średnią arytmetyczną.
3. Napisz program, który wczytuje ze standardowego wejścia liczbę wymierną i wypisuje ją na standardowym wyjściu w notacji wykładniczej (np.: 0.2 to **`2.0e-1`**).
4. Napisz program, który wczytuje ze standardowego wejścia liczbę wymierną i wypisuje ją na standardowym wyjściu z dokładnością do dwóch miejsc po przecinku.
5. Napisz program, który wczytuje ze standardowego wejścia trzy liczby całkowite i wypisuje na standardowym wyjściu największą z ich wartości (pamiętaj o przypadku gdy wszystkie podane liczby lub dwie z nich są równe).
6. Napisz program obliczający pole trójkąta na podstawie wymiarów podanych przez użytkownika. Użytkownik powinien mieć możliwość podania długości podstawy i wysokości lub wszystkich boków trójkąta.
7. Napisz program, który wczytuje ze standardowego wejścia współczynniki równania kwadratowego z jedną niewiadomą i wypisuje na standardowym

wyjściu wszystkie rozwiązania rzeczywiste tego równania lub odpowiednią informację w przypadku braku rozwiązań.

8. Napisz program, który w zależności od wyboru użytkownika wczytuje ze standardowego wejścia wymiary: kwadratu, prostokąta lub trójkąta i wypisuje na standardowym wyjściu pole figury o wczytanych wymiarach.

9. Napisz program kalkulator, który wykonuje wybraną przez użytkownika operacją arytmetyczną na dwóch wczytanych liczbach. Program powinien wczytywać dane ze standardowego wejścia i wypisywać wynik na standardowym wyjściu.

10. Napisz program wczytujący ze standardowego wejścia dwie dodatnie liczby całkowite n i m, i wypisujący w kolejnych wierszach na standardowym wyjściu wszystkie dodatnie wielokrotności n mniejsze od m.

11. Napisz program, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą n i wypisuje na standardowym wyjściu liczbę $n!$.

12. Napisz program, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą n i wypisuje na standardowym wyjściu element ciągu Fibonacciego o indeksie n.

13. Napisz program, który wczytuje ze standardowego wejścia dodatnie liczby całkowite n i m, i wypisuje na standardowym wyjściu największy wspólny dzielnik tych liczb.

14. Napisz program, który wczytuje ze standardowego wejścia nieujemną liczbę n i wypisuje na standardowym wyjściu wartość $\lfloor\sqrt{n}\rfloor$ zaokrągloną w dół do najbliższej wartości całkowitoliczbowej. Program napisz bez użycia funkcji z biblioteki matematycznej.

15. Napisz program, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą n i wypisuje na standardowym wyjściu wartość $0!+1!+\ldots+n!$.

16. Napisz program, który wczytuje ze standardowego wejścia liczbę całkowitą n i wypisuje na standardowe wyjście wartość bezwzględną z n. Do rozwiązania zadania nie używaj funkcji bibliotecznych za wyjątkiem operacji wejścia/wyjścia. W programie użyj samodzielnie zaimplementowanej funkcji liczącej wartość bezwzględną.

17. Napisz program, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą n i wypisuje na standardowym wyjściu liczbę $n!$. W programie użyj samodzielnie zaimplementowanej funkcji liczącej wartość silni.

18. Napisz funkcję, która dostaje jako argumenty nieujemne liczby całkowite n i m, z których co najmniej jedna jest różna od zera i zwraca jako wartość n^m.

19. Napisz program, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą n i wypisuje na standardowym wyjściu następującą sumę $\lfloor\sqrt{0}\rfloor+\lfloor\sqrt{1}\rfloor+\ldots+\lfloor\sqrt{n}\rfloor$ Algorytm wyliczania sumy podziel na dwie funkcje.

20. Napisz funkcję, która dostaje jako argument dodatnią liczbę całkowitą n i wypisuje na standardowym wyjściu wszystkie możliwe rozkłady liczby na sumy dwóch kwadratów dodatnich liczb całkowitych. Rozważ dwa przypadki:
    a) gdy a^2+b^2 i b^2+a^2 dla $a \neq b$ traktujemy jako dwa równe rozkłady,
    b) gdy a^2+b^2 i b^2+a^2 traktujemy jako ten sam rozkład i wypisujemy tylko jedne z nich.

    Jeżeli zajdzie taka potrzeba, możesz w rozwiązaniu używać funkcji pomocniczych.

21. Napisz funkcję, która wczytuje ze standardowego wejścia liczbę całkowitą i zwraca ją jako swoją wartość. Dodatkowo funkcja powinna sumować wszystkie dotychczas wczytane wartości i przy każdym swoim wywołaniu wypisywać na standardowym wyjściu ich aktualną sumę.

22. Napisz rekurencyjną funkcję, która dla otrzymanej w argumencie nieujemnej całkowitej liczby n zwraca jako wartość $n!$.

23. Napisz rekurencyjną funkcję zwracającą dla otrzymanej w argumencie nieujemnej liczby całkowitej n wartość elementu ciągu Fibonacciego o indeksie n.

24. Napisz rekurencyjną funkcję, która dostaje jako argumenty dwie dodatnie liczby całkowite n i m, i zwraca jako wartość największy wspólny dzielnik tych liczb obliczony algorytmem Euklidesa.

25. Napisz funkcję otrzymującą jako argumenty wskaźniki do dwóch zmiennych typu **`int`**, która zwraca jako wartość mniejszą z liczb wskazywanych przez argumenty.

26. Napisz funkcję otrzymującą jako argumenty wskaźniki do dwóch zmiennych typu **`int`**, która zwraca jako wartość wskaźnik na zmienną przechowującą mniejszą z liczb wskazywanych przez argumenty.

27. Napisz funkcję otrzymującą jako argumenty wskaźniki do dwóch zmiennych typu **`int`**, która zamienia ze sobą wartości wskazywanych zmiennych.

28. Napisz funkcję, której argumentami są **n** typu **int** oraz **w** wskaźnik do **int**, która przepisuje wartość **n** do zmiennej wskazywanej przez **w**.

29. Napisz bezargumentową funkcję, która rezerwuje pamięć dla pojedynczej zmiennej typu **int** i zwraca jako wartość wskaźnik do niej.

30. Napisz funkcję, która dostaje jako argument dodatnią liczbę całkowitą **n** i rezerwuje w pamięci blok **n** zmiennych typu **int** i zwraca jako wartość wskaźnik do początku zarezerwowanego bloku pamięci.

31. Napisz funkcję o dwóch argumentach:

    c) wskaźnik na funkcję o jednym argumencie typu **int** zwracającą wartość typu **double**,
    d) wartość typu **int**,

    która zwraca wartość funkcji otrzymanej w pierwszym argumencie na liczbie całkowitej podanej w drugim argumencie.

32. Napisz funkcję, która dostaje dwa argumenty: wskaźnik na stałą typu **int** i wskaźnik na zmienną typu **int**, i przepisuje zawartość stałej wskazywanej przez pierwszy argument do zmiennej wskazywanej przez drugi argument.

33. Napisz funkcję, która dostaje dwa argumenty: wskaźnik na stałą typu **int** i stały wskaźnik na zmienną typu **int**. Funkcja przepisuje zawartość stałej wskazywanej przez pierwszy argument do zmiennej wskazywanej przez drugi argument.

34. Napisz funkcję, która otrzymuje dwa argumenty: nieujemną liczbę całkowitą n oraz n-elementową tablicę tab elementów typu **int** i:

    a) nadaje wartość zero wszystkim elementom tablicy **tab**,
    b) zapisuje do kolejnych elementów tablicy wartości równe ich indeksom (do komórki o indeksie i funkcja ma zapisywać wartość **i**),
    c) podwaja wartość wszystkich elementów w tablicy **tab**,
    d) do wszystkich komórek tablicy tab wstawia wartości bezwzględne ich pierwotnych wartości.

35. Napisz funkcję, która otrzymuje dwa argumenty: dodatnią liczbę całkowitą n oraz n-elementową tablicę tab o elementach typu **const int** i zwraca jako wartość średnią arytmetyczną elementów tablicy **tab**.

36. Napisz funkcję, która otrzymuje jako argument liczbę całkowitą n ($n \geq 3$) i zwraca jako wartość największą liczbę pierwszą mniejszą od n (do wyznaczenia wyniku użyj algorytmu sita Eratostenesa).

37. Napisz funkcję, która otrzymuje trzy argumenty: dodatnią liczbę całkowitą n oraz dwie n-elementowe tablice **tab1**, **tab2** o elementach typu **int** i:

    e) przepisuje zawartość tablicy **tab1** do tablicy **tab2**,
    f) przepisuje zawartość tablicy **tab1** do tablicy **tab2** w odwrotnej kolejności (czyli element **tab1[0]** ma zostać zapisany do komórki tablicy **tab2** o indeksie $n-1$).

38. Napisz funkcję, która otrzymuje dwa argumenty: dodatnią liczbę całkowitą n oraz n-elementową tablicę tab o elementach typu **int** i:

    g) zwraca największą wartość przechowywaną w tablicy **tab**,
    h) zwraca najmniejszą wartość przechowywaną w tablicy **tab**,
    i) zwraca indeks elementu tablicy **tab** o największej wartości,
    j) zwraca indeks elementu tablicy **tab** o najmniejszej wartości,
    k) zwraca największą spośród wartości bezwzględnych elementów przechowywanych w tablicy **tab**,
    l) zwraca indeks elementu tablicy tab o największej wartości bezwzględnej.

39. Napisz funkcję, która otrzymuje jako argument dodatnią liczbę całkowitą n, a następnie tworzy dynamiczną n-elementową tablicę o elementach typu **int** i zwraca jako wartość wskaźnik do jej pierwszego elementu.

40. Napisz funkcję, która dostaje jako argument wskaźnik do jednowymiarowej dynamicznej tablicy o elementach typu **int** i zwalnia pamięć zajmowaną przez przekazaną w argumencie tablicę.

41. Napisz funkcję **wyczysc()**, która usuwa z tablicy przechowywany w niej napis (w sensie: umieszcza w niej poprawny napis o długości 0). Napisz dwie wersje funkcji **wyczysc()** dla napisów składających się ze znaków typu **char** i **wchar_t**.

42. Napisz funkcję **dlugosc()**, która jako argument otrzymuje napis i zwraca jako wartość jego długość. Napisz dwie wersje funkcji **dlugosc()** dla napisów składających się ze znaków typu **char** i **wchar_t**.

43. Napisz funkcję, która jako argumenty otrzymuje dwa napisy i zwraca wartość **1**, gdy pierwszy napis jest wcześniejszy w kolejności leksykograficznej i **0** w przeciwnym przypadku. Zakładamy, że oba napisy składają się ze znaków typu **char**, zawierają wyłącznie małe litery alfabetu łacińskiego, a system, na którym jest kompilowany i uruchamiany program, używa standardowego kodowania **ASCII**.

44. Napisz funkcję **kopiujn()**, która dostaje w argumentach dwie tablice znaków **nap1**, **nap2** oraz liczbę n i przekopiowuje n pierwszych znaków z napisu przechowywanego w tablicy **nap1** do tablicy **nap2**. W przypadku gdy

napis w tablicy **nap1** jest krótszy niż n znaków, funkcja powinna przepisać cały napis. Funkcja powinna zadbać o to, żeby na końcu napisu w tablicy nap2 znalazł się znak o kodzie **0**. Zakładamy, że w docelowej tablicy jest wystarczająco dużo miejsca. Napisz dwie wersje funkcji: dla napisów składających się ze znaków typu **char** i **wchar_t**.

45. Napisz funkcję sklej otrzymującą jako argumenty trzy tablice znaków i zapisującą do trzeciej tablicy konkatenację napisów znajdujących się w dwóch pierwszych tablicach. Zakładamy, że w trzeciej tablicy jest wystarczająco dużo miejsca. Napisz dwie wersje funkcji sklej dla napisów składających się ze znaków typu **char** i **wchar_t**.

46. Napisz funkcję **wytnijz()**, która dostaje jako argument dwa napisy **nap1** i **nap2**, i wycina z napisu **nap1** wszystkie znaki występujące w napisie **nap2**. Napisz dwie wersje funkcji dla napisów składających się ze znaków typu **char** i **wchar_t**.

47. Napisz funkcję, która wypisuje na standardowym wyjściu otrzymany w argumencie napis. Napisz dwie wersje funkcji: dla napisów składających się ze znaków typu **char** i **wchar_t**.

48. Napisz funkcję, która dostaje jako argument tablicę znaków i wczytuje do niej napis ze standardowego wejścia. Napisz dwie wersje funkcji: dla tablicy składających się ze znaków typu **char** i **wchar_t**.

49. Napisz funkcję godzina, która dostaje w argumentach trzy zmienne całkowite **godz**, **min** i **sek**, zawierające odpowiednio godziny, minuty oraz sekundy, i zwraca jako wartość napis zawierający godzinę w formacie **godz:min:sek**, w którym wartości poszczególnych pól pochodzą ze zmiennych podanych w argumentach. Napisz dwie wersje funkcji godzina: zwracające napisy będące tablicami znaków typu **char** i typu **wchar_t**.

50. Napisz funkcję, która dostaje jako argument dodatnie liczby całkowite n i m, tworzy dynamiczną dwuwymiarową tablicę tablic elementów typu **int** o wymiarach n na m, i zwraca jako wartość wskaźnik do niej.

51. Napisz funkcję, która dostaje jako argument dodatnie liczby całkowite n i m, tworzy dynamiczną dwuwymiarową tablicę elementów typu **int** o wymiarach n na m i zwraca jako wartość wskaźnik do niej.

52. Napisz funkcję, która dostaje jako argumenty wskaźnik do dwuwymiarowej tablicy tablic elementów typu **int** oraz jej wymiary: dodatnie liczby całkowite n i m, i usuwa z pamięci otrzymaną tablicę.

53. Napisz funkcję, która dostaje jako argumenty wskaźnik do tablicy dwuwymiarowej elementów typu **`int`** oraz jej wymiary wymiary n i m, i usuwa z pamięci otrzymaną tablicę.

54. Napisz funkcję, która dostaje w argumentach dwuwymiarową tablicę elementów typu **`int`**, o pierwszym wymiarze podanym jako drugi argument funkcji oraz drugim wymiarze równym **100** i wypełnia ją zerami.

55. Napisz funkcję, która dostaje w argumentach dwuwymiarową tablicę tablic elementów typu **`int`** oraz jej wymiary n i m, i wypełnia ją zerami.

56. Napisz funkcję, która dostaje w argumentach tablicę trójwymiarową o elementach typu **`int`** o wymiarach **100×100×100**, i zwraca jako wartość sumę wartości elementów tablicy.

57. Napisz funkcję, która dostaje jako argumenty dwuwymiarową tablicę tablic o elementach typu **`int`** oraz jej wymiary, i zwraca jako wartość indeks wiersza o największej średniej wartości elementów. Przyjmujemy, że dwa elementy leżą w tym samym wierszu, jeżeli mają taki sam pierwszy indeks.

58. Napisz funkcję, która dostaje jako argumenty dwuwymiarową tablicę tablic o elementach typu **`int`** oraz jej wymiary, i zwraca największą spośród średnich wartości elementów poszczególnych wierszy. Przyjmujemy, że dwa elementy leżą w tym samym wierszu, jeżeli mają taki sam pierwszy indeks.

59. Napisz funkcję, która dostaje jako argumenty dwuwymiarową kwadratową tablicę tablic tab o elementach typu **`int`** oraz jej wymiar, i zmienia kolejność elementów w otrzymanej tablicy w następujący sposób: dla dowolnych **`k`** i **`j`** element **`tab[k][j]`** ma zostać zamieniony miejscami z elementem **`tab[j][k]`.**

60. Napisz funkcję, która otrzymuje w argumentach dwie kwadratowe tablice tablic elementów typu **`int`** oraz ich wspólny wymiar, i zwraca jako wartość wynik mnożenia macierzy przechowywanych w przekazanych argumentach. Wynik powinien zostać zwrócony w nowo utworzonej tablicy tablic.

61. Napisz funkcję, która dostaje jako argument ścieżkę dostępu do pliku, otwiera plik do tekstowego czytania i zwraca jako wartość deskryptor świeżo otwartego pliku.

62. Napisz funkcję, która dostaje jako argument deskryptor do pliku tekstowego otwartego do czytania, wypisuje zawartość pliku na standardowe wyjście i zamyka plik.

63. Napisz funkcję, która dostaje jako argument ścieżkę dostępu do pliku tekstowego i wypisuje na standardowym wyjściu zawartość pliku z pominięciem białych znaków. Napisz dwie wersje funkcji dla znaków typów **`char`** i **`wchar_t`.**

64. Napisz funkcję, która dostaje jako argumenty ścieżkę dostępu do pliku tekstowego oraz znak **`c`** i zwraca jako wartość liczbę wystąpień znaku **`c`** w podanym w argumencie pliku.

65. Napisz funkcję, która dostaje jako argumenty ścieżki dostępu do dwóch plików tekstowych i zwraca jako wartość **1**, jeżeli podane pliki mają taką samą zawartość oraz **0** w przeciwnym wypadku. Napisz dwie wersje funkcji dla znaków typu **`char`** i **`wchar_t`**.

66. Napisz funkcję, która dostaje jako argumenty ścieżki dostępu do dwóch plików tekstowych i zwraca jako wartość **1**, jeżeli podane pliki mają taką samą zawartość z dokładnością do białych znaków oraz **0** w przeciwnym wypadku.

67. Napisz funkcję, która dostaje jako argumenty deskryptory dwóch plików tekstowych i przepisuje zawartość pierwszego pliku do drugiego pliku.

68. Napisz funkcję, która dostaje jako argumenty nazwę pliku, dwuwymiarową tablicę tablic o elementach typu **`int`** oraz wymiary tablicy i zapisuje binarnie zawartość tablicy do podanego pliku.

69. Napisz funkcję, która dostaje jako argumenty nazwę pliku, dwuwymiarową tablicę tablic o elementach typu **`int`** oraz wymiary tablicy i wczytuje binarnie zawartość pliku do tablicy. Napisz funkcję tak, aby była *kompatybilna* z funkcją z poprzedniego zadania.

7. Wskazówki

W rozwiązaniach zadań nie można używać instrukcji takich jak **`goto`**, **`continue`** czy **`break`** (za wyjątkiem wnętrza instrukcji switch), które są uważane za niezgodne z zasadami programowania strukturalnego. W języku C nie można przeciążać funkcji ani używać funkcji z domyślnymi wartościami argumentów. W niektórych zadaniach należy zaprojektować funkcję rekurencyjną, w wielu przypadkach ich użycie pozwala znacznie uprościć kod programu.

Należy pamiętać, że wskaźniki i tablice są ze sobą powiązane, przy czym w języku C możemy używać jednowymiarowych tablic nie wiedząc nic o wskaźnikach ani dynamicznym zarządzaniu pamięcią, wystarczy umiejętność deklarowania tablic automatycznych. Niektóre zadania będą wymagały operowania na wskaźnikach.

W języku C istnieją dwa rodzaje tablic dwuwymiarowych. Jeden rodzaj to tablice których elementami są tablice jednowymiarowe zaś drugi to tablice wskaźników do tablic jednowymiarowych. Możemy deklarować tablice różniące się sposobem utworzenia poszczególnych wymiarów. W zadaniach użyto określeń: **tablice tablic** (np. dla typu **`int **`**) i tablice wielowymiarowe (np. dla typów **`int[][n]`** lub **`nt[][n][m]`**).

Warto zapamiętać, że w przypadku tablic przekazanych do funkcji w jej argumentach, wewnątrz funkcji pracujemy na **oryginałach tablic**, a nie ich **kopiach**. Oznacza to, że zmiana wewnątrz funkcji wartości elementów takich tablic ma konsekwencje także na zewnątrz funkcji. Jest to zachowanie odmienne od argumentów funkcji o typach prostych, w przypadku których pracujemy na kopiach argumentów. Takie a nie inne zachowanie tablic wynika z faktu, że są one wskaźnikami na bloki pamięci. Skopiowany wskaźnik cały czas wskazuje na ten sam blok pamięci.

Napisy w języku C są przechowywane w jednowymiarowych tablicach o elementach typów znakowych. Istnieją dwa podstawowe typy znakowe **`char`** i **`wchar_t`**. Standard języka C nie określa długości tych typów. Typ **`char`** jest określony jako typ wystarczający do przechowywania podstawowego zestawu znaków na danym komputerze (w praktyce **`char`** jest typem jednobajtowym), zaś typ **`wchart_t`** powinien wystarczyć do przechowywania pełnego zestawu znaków dostępnego na danym komputerze. W zależności od implementacji **`wchar_t`** pozwala na kodowanie znaków w systemie **UTF-32** (w systemach GNU/Linux) lub **UTF-16** (w systemach Windows).

Na końcu poprawnego napisu w języku C (niezależnie od typu znaków z których się składa) znajduje się znak o kodzie **0**. Służy on do zaznaczenia końca napisu. Tablica przechowująca n-znakowy napis musi mieć co najmniej $n+1$ elementów, aby móc przechować znak kończący. Konieczność dbania o to, żeby na końcu napisu był zawsze znak kończący spoczywa na programiście.

Operacje na napisach o elementach typu **`char`** można znaleźć w bibliotece **`<string.h>`**. Typ **`wchar_t`** oraz funkcje operujące na zmiennych tego typu, odpowiadające operacjom z innych bibliotek standardowych (w tym z biblioteki **`<string.h>`**) znajdują się w bibliotece **`<wchar.h>`**. Funkcje z bibliotek standardowych wymagają dbania o to, żeby używane tablice znaków były wystarczających rozmiarów.

W języku C do operacji na plikach służą funkcje z biblioteki **`<stdio.h>`**, a wśród nich między innymi: **`fopen()`**, **`fclose()`**, **`fwrite()`**, **`fread()`**, **`fprintf()`**, **`fscanf()`**, **`feof()`** i **`fseek()`**. Otwierając plik należy określić w jakim celu plik jest otwierany (do czytania, do pisania etc.) oraz czy plik ma być otwarty w trybie binarnym (czyli jako ciąg bajtów) czy tekstowym.

Do wersji **C99** do obsługi wejścia i wyjścia należy używać tradycyjnej funkcji **`printf()`** i **`scanf()`** we wszystkich popularnych kompilatorach. W standardzie **C11** i późniejszych została wprowadzona możliwość implementacji tzw. bezpiecznych (*secure*) funkcji obsługi standardowego wejścia i wyjścia (np. zapewniające mechanizm ochrony przed przepełnieniem bufora). Przykładami są funkcje **`printf_s()`** i **`scanf_s()`**. Nie są one jednak obligatoryjne. Takie funkcje są dostępne w kompilatorze **MSVC** (firmy Microsoft) ale nie są dostępne w kolekcji **GCC** (projekt GNU).

W przypadku korzystania z MSVC należy korzystać z funkcji **`printf_s()`** i **`scanf_s()`** lub dodać dyrektywę: **`#define _CRT_SECURE_NO_WARNINGS`**.

8. Rozwiązania

Poniżej zaprezentowano rozwiązania większości zadań, które mogą sprawić trudność lub są pierwszymi zadaniami dotyczącymi konkretnego zagadnienia programowania (obsługa wejścia i wyjścia, wskaźniki, tablice etc.). Dość szczegółowy wykład dotyczący języka C (Koczubiej, 2025) można znaleźć pod adresem https://staff.tu.kielce.pl/sk/media/downloads/pp/pp-wyklad.pdf. Pomocne mogą też być opisy standardów języka C (C reference).

Zadanie nr 1.

```c
#include <stdio.h>
#include <math.h>

int main()
{
  double x;

  printf("Podaj liczbe wymierna x: ");
  scanf("%lf", &x);

  printf("Pierwiastek z liczby %f: %f", x, sqrt(x));

  printf("\n");
  return 0;
}
```

Listing 4. Rozwiązanie zadania nr 1

Zadanie nr 2.

```c
#include <stdio.h>

int main()
```

```c
{
  int l1, l2, l3;

  printf("Podaj liczbe calkowita l1: ");
  scanf("%d", &l1);

  printf("Podaj liczbe calkowita l2: ");
  scanf("%d", &l2);

  printf("Podaj liczbe calkowita l3: ");
  scanf("%d", &l3);

  printf("Srednia liczb: %f", (double)(l1 + l2 + l3) / 3);

  printf("\n");
  return 0;
}
```

Listing 5. Rozwiązanie zadania nr 2

Zadanie nr 4.

```c
#include <stdio.h>

int main()
{
  float f;

  printf("Podaj liczbe wymierna f: ");
  scanf("%f", &f);

  printf("Stala dokladnosc: %.2f", f);

  printf("\n");
  return 0;
}
```

Listing 6. Rozwiązanie zadania nr 4

Zadanie nr 8.

```c
#include <stdio.h>

int main()
{
  int i;
  double a, b, h, p;

  printf("Wybierz figure:\n");
  printf("  1. Kwadrat\n");
  printf("  2. Prostokat \n");
```

```c
  printf("  3. Trojkat\n");

  scanf("%d", &i);

  switch(i)
  {
    case 1:
      printf("Podaj bok kwadratu: ");
      scanf("%lf", &a);
      p = a * a;
      break;

    case 2:
      printf("Podaj boki prostokata: ");
      scanf("%lf %lf", &a, &b);
      p = a * b;
      break;

    case 3:
      printf("Podaj podstawe i wysokosc trojkata: ");
      scanf("%lf %lf", &a, &h);
      p = 0.5 * a * h;
  }

  printf("Pole figury: %f", p);

  printf("\n");
  return 0;
}
```

Listing 7. Rozwiązanie zadania nr 8

Zadanie nr 10.

```c
#include <stdio.h>

int main()
{
  int i, n, m;

  printf("Podaj liczbe calkowita n: ");
  scanf("%d", &n);

  printf("Podaj liczbe calkowita m: ");
  scanf("%d", &m);

  printf("Kolejne wielokrotnosci %d:\n", n);
  for(i = n; i < m; i += n)
    printf("%d\n", i);
```

```
  printf("\n");
  return 0;
}
```

Listing 8. Rozwiązanie zadania nr 10

Zadanie nr 11.

```
#include <stdio.h>

int main()
{
  int n, i, sil = 1;

  printf("Podaj calkowita n: ");
  scanf("%d", &n);

  for(i = 2; i <= n; i++)
    sil *= i;

  printf("%d! = %d", n, sil);

  printf("\n");
  return  0;
}
```

Listing 9. Rozwiązanie zadania nr 11

Zadanie nr 12.

```
#include <stdio.h>

int main()
{
  int n, i, tmp, fib1 = 1, fib2 = 1;

  printf("Podaj liczbe calkowita n: ");
  scanf("%d", &n);

  for(i = 2; i <= n; i++)
  {
    tmp = fib1;
    fib1 = fib2 + fib1;
    fib2 = tmp;
  }

  printf("%d element ciagu: %d", n, fib1);

  printf("\n");
```

```
  return 0;
}
```

Listing 10. Rozwiązanie zadania nr 12

Zadanie nr 13.

```c
#include <stdio.h>

int main()
{
  int i, n, m, max, nwd = 1;

  printf("Podaj liczbe calkowita n: ");
  scanf("%d", &n);

  printf("Podaj liczbe calkowita m: ");
  scanf("%d", &m);

  max = (n > m) ? n : m;

  for(i = 2; i <= max;  i++)
    if((n % i == 0) && (m % i == 0)) nwd = i;

  printf("NWD(%d, %d) = %d", n, m, nwd);

  printf("\n");
  return 0;
}
```

Listing 11. Rozwiązanie zadania nr 13

Zadanie nr 14.

```c
#include <stdio.h>

int main()
{
  int x, i, pier = 0;

  printf("Podaj liczbe calkowita x: ");
  scanf("%d", &x);

  for(i = 1; i <= x; i++)
    if(i * i <= x)
      pier = i;

  printf("Przyblizony pierwiastek z liczby %d: %d", x, pier);

  printf("\n");
```

```
  return 0;
}
```

Listing 12. Rozwiązanie zadania nr 14

Zadanie nr 15.

```c
#include <stdio.h>

int main()
{
  int i, n, sil = 1, sum = 1;

  printf("Podaj liczbe calkowita n: ");
  scanf("%d", &n);

  for(i = 1; i <= n; i++)
  {
    sil *= i;
    sum += sil;
  }

  printf("0! + 1! + ... + %d! = %d", n, sum);

  printf("\n");
  return  0;
}
```

Listing 13. Rozwiązanie zadania nr 15

Zadanie nr 16.

```c
#include <stdio.h>

int bezwzgledna(int l)
{
  if(l < 0)
    return (-1) * l;
  else
    return l;
}

int main()
{
  int n;

  printf("Podaj liczbe calkowita: ");
  scanf("%d", &n);

  printf("|%d| = %d\n", n, bezwzgledna(n));
```

```
  printf("\n");
  return 0;
}
```

Listing 14. Rozwiązanie zadania nr 16

Zadanie nr 17.

```
#include <stdio.h>

int silnia(int l)
{
  int i, s = 1;

  for(i = 2; i <= l; i++)
    s *= i;

  return s;
}

int main()
{
  int n;

  printf("Podaj liczbe calkowita n: ");
  scanf("%u", &n);

  printf("%d! = %d", n, silnia(n));

  printf("\n");
  return 0;
}
```

Listing 15. Rozwiązanie zadania nr 17

Zadanie nr 20a.

```
#include <stdio.h>

int pierw(int x)
{
  int s, p = 0, k= x;

  while(k - p > 1)
  {
    s = (p + k) / 2;

    if(s * s <= x)
      p = s;
    else
      k = s;
```

```
  }

  if(x <= 1)
    return k;
  else
    return p;
}

void wyp(int n)
{
  int i, p;

  for(i = 1; i <= pierw(n); i++)
  {
    p = pierw(n - i * i);

    if((p != 0) && (i * i + p * p == n))
      printf("%d * %d + %d * %d = %d\n", i, i, p, p, n);
  }
}

int main()
{
  int n;

  printf("podaj liczbe calkowita n: ");
  scanf("%d", &n);

  printf("rozklady liczby %u:\n", n);
  wyp(n);

  printf("\n");
  return 0;
}
```

Listing 16. Rozwiązanie zadania nr 20a

Zadanie nr 20b

```
#include <stdio.h>

int pierw(int x)
{
  int s, p = 0, k = x;

  while (k - p > 1)
  {
    s = (p + k) / 2;

    if (s * s <= x)
```

```
      p = s;
    else
      k = s;
  }

  if (x <= 1)
    return k;
  else
    return p;
}

void wyp(int n)
{
  int i, p;

  for (i = 1; i <= pierw(n); i++)
  {
    p = pierw(n - i * i);

    if ((i <= p) && (i * i + p * p == n))
      printf("%d * %d + %d * %d = %d\n", i, i, p, p, n);
  }
}

int main()
{
  int n;

  printf("podaj liczbe calkowita n: ");
  scanf("%u", &n);

  printf("rozklady liczby %d:\n", n);
  wyp(n);

  printf("\n");
  return 0;
}
```

Listing 17. Rozwiązanie zadania nr 20b

Zadanie nr 22.

```
#include <stdio.h>

int silnia(int n)
{
  if(n <= 1)
    return 1;
  else
    return silnia(n - 1) * n;
```

```c
}

int main()
{
  int n;

  printf("Podaj liczbe calkowita n: ");
  scanf("%u", &n);

  printf("%d! = %d", n, silnia(n));

  printf("\n");
  return 0;
}
```

Listing 18. Rozwiązanie zadania nr 22

Zadanie nr 23.

```c
#include <stdio.h>

int fib(int n)
{
  if(n <= 1)
    return 1;
  else
    return fib(n - 1) + fib(n - 2);
}

int main()
{
  int n;

  printf("Podaj liczbe calkowita n: ");
  scanf("%u", &n);

  printf("%u element ciagu: %u", n, fib(n));

  printf("\n");
  return 0;
}
```

Listing 19. Rozwiązanie zadania nr 23

Zadanie nr 24.

```c
#include <stdio.h>

int nwd(int n, int m)
{
  if (n == m)
```

```
    return m;

  if (n > m)
    return nwd(n - m, m);
  else
    return nwd(m - n, n);
}

int nwd2(int n, int m)
{
  if (m != 0)
    return nwd(m, n % m);

  return n;
}

int main()
{
  int n, m;

  printf("Podaj liczbe calkowita n: ");
  scanf("%u", &n);

  printf("Podaj liczbe calkowita m: ");
  scanf("%u", &m);

  printf("NWD(%u, %u) = %u\n", n, m, nwd(n, m));
  printf("NWD2(%u, %u) = %u", n, m, nwd2(n, m));

  printf("\n");
  return 0;
}
```

Listing 20. Rozwiązanie zadania nr 24

Zadanie nr 25.

```
#include <stdio.h>
#include <stdlib.h>

int mniejsza(int *a, int *b)
{
  if(*a < *b)
    return *a;
  else
    return *b;
}

int mniejsza2(int *a, int *b)
{
```

```c
  return (*a < *b) ? *a : *b;
}

int main()
{
  int *n = malloc(sizeof *n);
  int *m = malloc(sizeof *m);

  int q, w;

  printf("Podaj liczbe calkowita n: ");
  scanf("%d", n);

  printf("Podaj liczbe calkowita m: ");
  scanf("%d", m);

  q = *n;
  w = *m;

  printf("MIN(%d, %d) = %d\n", *n, *m, mniejsza(n, m));
  printf("MIN(%d, %d) = %d", q, w, mniejsza2(&q, &w));

  free(n);
  free(m);

  printf("\n");
  return 0;
}
```

Listing 21. Rozwiązanie zadania nr 25

Zadanie nr 26.

```c
#include <stdio.h>
#include <stdlib.h>

int * mniejsza(int *a, int *b)
{
  if (*a < *b)
    return a;
  else
    return b;
}

int main()
{
  int *n = malloc(sizeof *n);
  int *m = malloc(sizeof *m);

  printf("Podaj liczbe calkowita n: ");
```

```
  scanf("%d", n);

  printf("Podaj liczbe calkowita m: ");
  scanf("%d", m);

  printf("MIN(%d, %d) = %d\n", *n, *m, *mniejsza(n, m));

  free(n);
  free(m);

  printf("\n");
  return 0;
}
```

Listing 22. Rozwiązanie zadania nr 26

Zadanie nr 27.

```
#include <stdio.h>
#include <stdlib.h>

void zamien(int *a, int *b)
{
  int tmp;

  tmp = *a;
  *a = *b;
  *b = tmp;
}

int main()
{
  int *n = malloc(sizeof *n);
  int *m = malloc(sizeof *m);

  printf("Podaj liczbe calkowita n: ");
  scanf("%d", n);

  printf("Podaj liczbe calkowita m: ");
  scanf("%d", m);

  zamien(n, m);

  printf("Wartosc liczby n: %d\n", *n);
  printf("Wartosc liczby m: %d", *m);

  free(n);
  free(m);

  printf("\n");
```

```
  return 0;
}
```

Listing 23. Rozwiązanie zadania nr 27

Zadanie nr 29.

```
#include <stdio.h>
#include <stdlib.h>

int * alokuj()
{
  return malloc(sizeof(int));
}

int main()
{
  int *n = alokuj();

  printf("Podaj liczbe calkowita n: ");
  scanf("%d", n);

  printf("Adres liczby: %p\n", (void *)n);
  printf("Wartosc liczby n: %d", *n);

  free(n);

  printf("\n");
  return 0;
}
```

Listing 24. Rozwiązanie zadania nr 29

Zadanie nr 30.

```
#include <stdio.h>
#include <stdlib.h>

int * alokuj(int n)
{
  return malloc(n * sizeof(int));
}

int main()
{
  int n;
  int i, *p;

  printf("Podaj liczbe calkowita n: ");
  scanf("%u", &n);
```

```
  p = alokuj(n);
  printf("Adres bloku: %p\n", (void *)p);

  for(i = 0; i < n; i++)
    *(p + i) = 2 * i;

  printf("Wartosc liczb w bloku:\n");
  for(i = 0; i < n; i++)
    printf("[%d]: %d\n", i, *(p + i));

  free(p);

  printf("\n");
  return 0;
}
```

Listing 25. Rozwiązanie zadania nr 30

Zadanie nr 31.

```
#include <stdio.h>
#include <math.h>

double pierw(int x)
{
  return sqrt(x);
}

double wywolaj(double (* fun)(int arg), int akt)
{
  return fun(akt);
}

int main()
{
  int n;
  double y;

  printf("Podaj liczbe calkowita n: ");
  scanf("%d", &n);

  y = wywolaj((* pierw), n);

  printf("Liczba y: %f", y);

  printf("\n");
  return 0;
}
```

Listing 26. Rozwiązanie zadania nr 31

Zadanie nr 33.

```c
#include <stdio.h>

void przepisz(int const *a, int *const b)
{
  *b = *a;
}

int main()
{
  int xx = 13;
  int yy = 69;

  int const *x = &xx;
  int *const y = &yy;

  printf("Wartosc liczby xx: %d\n", xx);
  printf("Wartosc liczby yy: %d\n", yy);

  printf("Wartosc liczby x: %d\n", *x);
  printf("Adres liczby x: %p\n", x);

  printf("Wartosc liczby y: %d\n", *y);
  printf("Adres liczby y: %p\n\n", y);

  przepisz(x, y);

  printf("Wartosc liczby xx: %d\n", xx);
  printf("Wartosc liczby yy: %d\n", yy);

  printf("Wartosc liczby x: %d\n", *x);
  printf("Adres liczby x: %p\n", x);

  printf("Wartosc liczby y: %d\n", *y);
  printf("Adres liczby y: %p\n", y);

  printf("\n");
  return 0;
}
```

Listing 27. Rozwiązanie zadania nr 33

Zadanie nr 34.

```c
#include <stdio.h>

void zeruj(int n, int* t)
{
  int i;
```

```
  for(i = 0; i < n; i++)
    t[i] = 0;
}

void inicjuj(int n, int* t)
{
  int i;

  for(i = 0; i < n; i++)
    t[i] = i;
}

int main()
{
  int i;
  int t[] = {11, 12, 13, 14, 15};
  int n = sizeof(t) / sizeof(t[0]);

  printf("Tablica po deklaracji: ");
  for(i = 0; i < n; i++)
    printf("[%d]=%d%s", i, t[i], (i < n - 1)?", ":"\n");

  zeruj(n, t);

  printf("Tablica po zeruj(): ");
  for(i = 0; i < n; i++)
    printf("[%d]=%d%s", i, t[i], (i < n - 1)?", ":"\n");

  inicjuj(n, t);

  printf("Tablica po inicjuj(): ");
  for(i = 0; i < n; i++)
    printf("[%d]=%d%s", i, t[i], (i < n - 1)?", ":"\n");

  printf("\n");
  return 0;
}
```

Listing 28. Rozwiązanie zadania nr 34

Zadanie nr 35.

```
#include <stdio.h>

double srednia(int n, const int *t)
{
  int i;
  double sr = 0;

  for(i = 0; i < n; i++)
```

```
    sr += t[i];

  sr /= n;
  return sr;
}

int main()
{
  double s = 0;

  int t[] = {11, 12, 13, 14, 15};
  int n = sizeof(t) / sizeof(t[0]);

  s = srednia(n, t);

  printf("Liczba s: %f", s);

  printf("\n");
  return 0;
}
```

Listing 29. Rozwiązanie zadania nr 35

Zadanie nr 36.

```
#include <stdio.h>
#include <stdbool.h>

int pierwsza(int n)
{
  int i, j, tmp;
  bool sito[n];
//  bool * sito = malloc(n * sizeof(bool));

  for(i = 0; i < n; i++)
    sito[i] = true;
  for(i = 2; i < n; i++)
    if(sito[i])
    {
      tmp = i;
      for(j = 2 * i; j < n; j += i)
        sito[j] = false;
    }
//  free(sito);

  return tmp;
}

int main()
{
```

```
  int n, y;

  printf("Podaj liczbe calkowita n >= 3: ");
  scanf("%d", &n);

  y = pierwsza(n);

  printf("Liczba y: %d", y);

  printf("\n");
  return 0;
}
```

Listing 30. Rozwiązanie zadania nr 36

Zadanie nr 37b.

```
#include <stdio.h>

void przepisz(int n, int *tab1, int *tab2)
{
  int i;

  for(i = 0; i < n; i++)
    tab2[i] = tab1[i];
}

int main()
{
  int i;

  int t1[] = {11, 12, 13, 14, 15};
  int t2[] = { 0,  0,  0,  0,  0};
  int n = sizeof(t2) / sizeof(t2[0]);

  printf("Tablica po deklaracji: ");
  for(i = 0; i < n; i++)
    printf("[%d]=%d%s", i, t2[i], (i < n - 1)?", ":"\n");

  przepisz(n, t1, t2);

  printf("Tablica po przepisz(): ");
  for(i = 0; i < n; i++)
    printf("[%d]=%d%s", i, t2[i], (i < n - 1)?", ":"\n");

  printf("\n");
  return 0;
}
```

Listing 31. Rozwiązanie zadania nr 37b

Zadanie nr 38a.

```c
#include <stdio.h>

int maksimum(int n, int *tab)
{
  int i, max = tab[0];

  for(i = 1; i < n; i++)
    if(tab[i] > max)
      max = tab[i];

  return max;
}

int main()
{
  int m;

  int t[] = {11, 12, 69, 14, 15};
  int n = sizeof(t) / sizeof(t[0]);

  m = maksimum(n, t);

  printf("Wartosc maksymalna m: %d", m);

  printf("\n");
  return 0;
}
```

Listing 32. Rozwiązanie zadania nr 38a

Zadanie nr 38c.

```c
#include <stdio.h>

int maksimum(int n, int *tab)
{
  int i, max = 0;

  for(i = 1; i < n; i++)
    if(tab[i] > tab[max])
      max = i;

  return max;
}

int main()
{
  int m;
```

```
  int t[] = {11, 12, 69, 14, 15};
  int n = sizeof(t) / sizeof(t[0]);

  m = maksimum(n, t);

  printf("Polozenie wartosci maksymalnej m: %d", m);

  printf("\n");
  return 0;
}
```

Listing 33. Rozwiązanie zadania nr 38c

Zadanie nr 39.

```
#include <stdio.h>
#include <stdlib.h>

int * alokuj(int n)
{
  return malloc(n * sizeof(int));
}

int main()
{
  int *t, i, n = 3;

  t = alokuj(n);
  t[0] = 0, t[1] = 13, t[2] = 0;

  printf("Tablica: ");
  for(i = 0; i < n; i++)
    printf("[%d]=%d%s", i, t[i], (i < n - 1)?", ":"\n");

  free(t);

  printf("\n");
  return 0;
}
```

Listing 34. Rozwiązanie zadania nr 39

Zadanie nr 40.

```
#include <stdio.h>
#include <stdlib.h>

void zwolnij(int *tab)
{
  free(tab);
}
```

```c
int main()
{
  int *t, i, n = 3;
  t = malloc(n * sizeof(int));

  t[0] = 0, t[1] = 13, t[2] = 0;

  printf("Tablica: ");
  for(i = 0; i < n; i++)
    printf("[%d]=%d%s", i, t[i], (i < n - 1)?", ":"\n");

  zwolnij(t);

  printf("\n");
  return 0;
}
```

Listing 35. Rozwiązanie zadania nr 40

Zadanie nr 41.

```c
#include <stdio.h>

void wyczysc(char *nap)
{
  nap[3] = 0;
}

int main()
{
  char s[] = "ala ma psa";

  printf("s: %s\n", s);

  wyczysc(s);

  printf("s po wyczysc(): %s", s);

  printf("\n");
  return 0;
}
```

Listing 36. Rozwiązanie zadania nr 41

Zadanie nr 42.

```c
#include <stdio.h>

int dlugosc(char *nap)
{
```

```
  int i = 0;

  while(nap[i] != 0)
    i++;

  return i;
}

int main()
{
  int n;
  char s[] = "ala ma psa";

  n = dlugosc(s);

  printf("Dlugosc napisu n: %d", n);

  printf("\n");
  return 0;
}
```

Listing 37. Rozwiązanie zadania nr 42

Zadanie nr 43.

```
#include <stdio.h>

int porownaj(char *nap1, char *nap2)
{
  int i;

  for(i = 0; (nap1[i] != 0) && (nap2[i] != 0); i++)
    if(nap1[i] != nap2[i])
      return (nap1[i] < nap2[i]) ? 1 : 2;

  return (nap1[i] != 0) ? 1 : 2;
}

int main()
{
  int n;
  char s1[] = "ala ma psa";
  char s2[] = "ala ma jeza";

  n = porownaj(s1, s2);

  printf("Wartosc n: %d", n);

  printf("\n");
```

```
  return 0;
}
```

Listing 38. Rozwiązanie zadania nr 43

Zadanie nr 45.

```
#include <stdio.h>

void sklej(char *nap1, char *nap2, char *nap3)
{
  int i, j;

  for(i = 0; nap1[i] != 0; i++)
    nap3[i] = nap1[i];

  for(j = 0; nap2[j] != 0; i++, j++)
    nap3[i] = nap2[j];

  nap3[i] = 0;
}

int main()
{
  int n;
  char s1[] = "ala ma psa";
  char s2[] = " a ola ma jeza";
  char s3[80];

  sklej(s1, s2, s3);

  printf("Napis s3: %s", s3);

  printf("\n");
  return 0;
}
```

Listing 39. Rozwiązanie zadania nr 45

Zadanie nr 46.

```
#include <stdio.h>

void wytnijz(char *nap1, char *nap2)
{
  int i, j;
  int wyst[256] = {};

  for(i = 0; nap2[i] != 0; i++)
    wyst[nap2[i]] = 1;
```

```
  for(j = 0, j = 0; nap1[i] != 0; i++)
    if(wyst[nap1[i]] == 0)
    {
      if(j < i)
        nap1[j] = nap1[i];
      j++;
    }
  nap1[j] = 0;
}

int main()
{
  int n;
  char s1[] = "ala ma psa";
  char s2[] = "ms";

  wytnijz(s1, s2);

  printf("Napis s1: %s", s1);

  printf("\n");
  return 0;
}
```

Listing 40. Rozwiązanie zadania nr 46

Zadanie nr 47.

```
#include <stdio.h>

void wypisz(char *nap)
{
  printf("%s", nap);
}

int main()
{
  char s[] = "ala ma psa";

  wypisz(s);

  printf("\n");
  return 0;
}
```

Listing 41. Rozwiązanie zadania nr 47

Zadanie nr 48.

```
#include <stdio.h>
#include <string.h>
```

```c
void wczytaj(char *nap)
{
//  scanf("%79[^\n]", nap);

  fgets(nap, 80, stdin);

//  if((strlen(nap) > 0) && (nap[strlen(nap) - 1] == '\n'))
//    nap[strlen(nap) - 1] = '\0';
}

int main()
{
  char s[80];

  wczytaj(s);

  printf("Napis s: %s", s);

  printf("\n");
  return 0;
}
```

Listing 42. Rozwiązanie zadania nr 48

Zadanie nr 49.

```c
#include <stdio.h>
#include <stdlib.h>

char * godzina(int godz, int min, int sek)
{
  char *nap = malloc(9 * sizeof(char));

  sprintf(nap, "%02d:%02d:%02d", godz, min, sek);

  return nap;
}

int main()
{
  char *n;
  int g = 12, m = 9, s = 7;

  n = godzina(g, m, s);

  printf("Napis n: %s", n);

  free(n);
```

```
  printf("\n");
  return 0;
}
```

Listing 43. Rozwiązanie zadania nr 49

Zadanie nr 50.

```
#include <stdio.h>
#include <stdlib.h>

int ** alokuj(int n, int m)
{
  int i;

  int **tab = malloc(n * sizeof(int *));

  for(i = 0; i < n; i++)
    tab[i] = malloc(m * sizeof(int));

  return tab;
}

int main()
{
  int i, j, n = 4, m = 3, **t;

  t = alokuj(n, m);

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      t[i][j] = i + j;

  printf("Wartosci w t:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  for(i = 0; i < n; i++)
    free(t[i]);
  free(t);

  printf("\n");
  return 0;
}
```

Listing 44. Rozwiązanie zadania nr 50

Zadanie nr 51.

```c
#include <stdio.h>
#include <stdlib.h>

int (* alokuj(int n, int m))[]
{
  return malloc(n * sizeof(int[m]));
}

int main()
{
  int i, j, n = 4, m = 3, (* t)[m];

  t = alokuj(n, m);

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      t[i][j] = i + j;

  printf("Wartosci w t:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  free(t);

  printf("\n");
  return 0;
}
```

Listing 45. Rozwiązanie zadania nr 51

Zadanie nr 52.

```c
#include <stdio.h>
#include <stdlib.h>

void zwolnij(int n, int m, int **tab)
{
  int i;

  for(i = 0; i < n; i++)
    free(tab[i]);

  free(tab);
}

int main()
{
```

```c
  int i, j, n = 4, m = 3, **t;

  t = malloc(n * sizeof(int *));
  for(i = 0; i < n; i++)
    t[i] = malloc(m * sizeof(int));

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      t[i][j] = i + j;

  printf("Wartosci w t:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  zwolnij(n, m, t);

  printf("\n");
  return 0;
}
```

Listing 46. Rozwiązanie zadania nr 52

Zadanie nr 53.

```c
#include <stdio.h>
#include <stdlib.h>

void zwolnij(int n, int m, int (* tab)[m])
{
  free(tab);
}

int main()
{
  int i, j, n = 4, m = 3, (* t)[m];

  t = malloc(n * sizeof(int[m]));;

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      t[i][j] = i + j;

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  zwolnij(n, m, t);
```

```
  printf("\n");
  return 0;
}
```

Listing 47. Rozwiązanie zadania nr 53

Zadanie nr 54.

```
#include <stdio.h>
#define TS 5

void zeruj(int tab[][TS], int n)
{
  int i, j;

  for(i = 0; i < n; i++)
    for(j = 0; j < TS; j++)
      tab[i][j] = 7;
}

int main()
{
  int i, j, n = TS, m = TS, t[TS][TS];

  zeruj(t, n);

  printf("Wartosci w t:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  printf("\n");
  return 0;
}
```

Listing 48. Rozwiązanie zadania nr 54

Zadanie nr 55.

```
#include <stdio.h>
#include <stdlib.h>
#define TS 5

void zeruj(int **tab, int n, int m)
{
  int i, j;

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      tab[i][j] = 7;
```

```c
}

int main()
{
  int i, j, n = TS, m = TS, **t;

  t = malloc(n * sizeof(int *));
  for(i = 0; i < n; i++)
    t[i] = malloc(m * sizeof(int));

  zeruj(t, n, m);

  printf("Wartosci w t:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  for(int i = 0; i < n; i++)
    free(t[i]);
  free(t);

  printf("\n");
  return 0;
}
```

Listing 49. Rozwiązanie zadania nr 55

Zadanie nr 56.

```c
#include <stdio.h>
#define TS 5

int suma(int tab[][TS][TS])
{
  int i, j, k, sum = 0;

  for(i = 0; i < TS; i++)
    for(j = 0; j < TS; j++)
      for(k = 0; k < TS; k++)
        sum += tab[i][j][k];

  return sum;
}

int main()
{
  int i, j, k, t[TS][TS][TS], s;

  for(i = 0; i < TS; i++)
```

```
    for(j = 0; j < TS; j++)
      for(k = 0; k < TS; k++)
        t[i][j][k] = 1;

  s = suma(t);

  printf("Wartosc sumy s: %d\n", s);

  printf("\n");
  return 0;
}
```

Listing 50. Rozwiązanie zadania nr 56

Zadanie nr 57.

```
#include <stdio.h>
#include <stdlib.h>
#define TS 5

int maxsred(int **tab, int n, int m)
{
  int i, j, sum, max;
  double war;

  for(i = 0; i < n; i++)
  {
    sum = 0;

    for(j = 0; j < m; j++)
      sum += tab[i][j];

    if(((double)sum / m > war) || (i == 0))
    {
      max = i;
      war = (double)sum / m;
    }
  }

  return max;
}

int main()
{
  int i, j, n = TS, m = TS, **t, ind;

  t = malloc(n * sizeof(int *));
  for(i = 0; i < n; i++)
    t[i] = malloc(m * sizeof(int));
```

```
  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      t[i][j] = i + j;

  printf("Wartosci w t:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  ind = maxsred(t, n, m);

  printf("\nLiczba ind: %d", ind);

  for(int i = 0; i < n; i++)
    free(t[i]);
  free(t);

  printf("\n");
  return 0;
}
```

Listing 51. Rozwiązanie zadania nr 57

Zadanie nr 58.

```
#include <stdio.h>
#include <stdlib.h>
#define TS 5

double maxsred(int **tab, int n, int m)
{
  int i, j, sum;
  double war;

  for(i = 0; i < n; i++)
  {
    sum = 0;

    for(j = 0; j < m; j++)
      sum += tab[i][j];

    if(((double)sum / m > war) || (i == 0))
      war = (double)sum / m;
  }

  return war;
}

int main()
```

```c
{
  int i, j, n = TS, m = TS, **t;
  double sr;

  t = malloc(n * sizeof(int *));
  for(i = 0; i < n; i++)
    t[i] = malloc(m * sizeof(int));

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      t[i][j] = i + j;

  printf("Wartosci w t:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  sr = maxsred(t, n, m);

  printf("\nLiczba sr: %f", sr);

  for(int i = 0; i < n; i++)
    free(t[i]);
  free(t);

  printf("\n");
  return 0;
}
```

Listing 52. Rozwiązanie zadania nr 58

Zadanie nr 60.

```c
#include <stdio.h>
#include <stdlib.h>
#define TS 5

int ** pomnoz(int n, int **tab1, int **tab2)
{
  int i, j, k, **tab3;

  tab3 = malloc(n * sizeof(int *));
  for(i = 0; i < n; i++)
    tab3[i] = malloc(n * sizeof(int));

  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
    {
      tab3[i][j] = 0;
```

```
      for(k = 0; k < n; k++)
        tab3[i][j] += tab1[i][k] * tab2[k][j];
    }

  return tab3;
}

int main()
{
  int i, j, n = TS, **t1, **t2, **t3;

  t1 = malloc(n * sizeof(int *));
  for(i = 0; i < n; i++)
    t1[i] = malloc(n * sizeof(int));

  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
      t1[i][j] = i + j;

  t2 = malloc(n * sizeof(int *));
  for(i = 0; i < n; i++)
    t2[i] = malloc(n * sizeof(int));

  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
      t2[i][j] = 2 * i + j;

  printf("Wartosci w t1:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
      printf("[%d,%d]=%d%s", i, j, t1[i][j], (j < n - 1)?", ":"\n");

  printf("Wartosci w t2:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
      printf("[%d,%d]=%d%s", i, j, t2[i][j], (j < n - 1)?", ":"\n");

  t3 =  pomnoz(n, t1, t2);

  printf("\nWartosci w t3:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
      printf("[%d,%d]=%d%s", i, j, t3[i][j], (j < n - 1)?", ":"\n");

  for(int i = 0; i < n; i++)
    free(t1[i]);
  free(t1);
```

```
  for(int i = 0; i < n; i++)
    free(t2[i]);
  free(t2);

  for(int i = 0; i < n; i++)
    free(t3[i]);
  free(t3);

  printf("\n");
  return 0;
}
```

Listing 53. Rozwiązanie zadania nr 60

Zadanie nr 61.

```
#include <stdio.h>
#include <stdlib.h>

FILE * otworz(char * nazwa)
{
  return fopen(nazwa, "rt");
}

int main()
{
  char nazwa[] = "plik.txt";
  FILE *plik;
  int wyn;

  plik = otworz(nazwa);
  printf("Deskryptor: %p\n", plik);

  if(!plik)
  {
    printf("Blad otwierania: %s.\n", nazwa);
    return EXIT_FAILURE;
  }

  wyn = fclose(plik);

  if(wyn)
  {
    printf("Blad zamykania: %s.\n", nazwa);
    return EXIT_FAILURE;
  }

  printf("\n");
```

```
  return 0;
}
```

Listing 54. Rozwiązanie zadania nr 61

Zadanie nr 62.

```
#include <stdio.h>
#include <stdlib.h>

void wypisz(FILE * plik)
{
  char c;

//  while(fscanf(plik, "%c", &c) == 1)
//    printf("%c", c);

//  while(!feof(plik))
//  {
//    c = fgetc(plik);
//    putchar(c);
//  }

  while((c = fgetc(plik)) != EOF)
  {
    putchar(c);
  }
}

int main()
{
  char nazwa[] = "plik.txt";
  FILE *plik;
  int wyn;

  plik = fopen(nazwa, "rt");
  printf("Deskryptor: %p\n", plik);

  if(!plik)
  {
    printf("Blad otwierania: %s.\n", nazwa);
    return EXIT_FAILURE;
  }

  wypisz(plik);

  wyn = fclose(plik);

  if(wyn)
  {
```

```
    printf("Blad zamykania: %s.\n", nazwa);
    return EXIT_FAILURE;
  }

  printf("\n");
  return 0;
}
```

Listing 55. Rozwiązanie zadania nr 62

Zadanie nr 63.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

void wypisz(char * nazwa)
{
  FILE *plik;
  int wyn;
  char c;

  plik = fopen(nazwa, "rt");
  printf("Deskryptor: %p\n", plik);

  if(!plik)
  {
    printf("Blad otwierania: %s.\n", nazwa);
    exit(EXIT_FAILURE);
  }

  while((c = fgetc(plik)) != EOF)
  {
    if(!isspace(c))
      putchar(c);
  }

  wyn = fclose(plik);

  if(wyn)
  {
    printf("Blad zamykania: %s.\n", nazwa);
    exit(EXIT_FAILURE);
  }
}

int main()
{
  char nazwa[] = "plik.txt";
```

```
  wypisz(nazwa);

  printf("\n");
  return 0;
}
```

Listing 56. Rozwiązanie zadania nr 63

Zadanie nr 64.

```
#include <stdio.h>
#include <stdlib.h>

int wyst(char * nazwa, char z)
{
  FILE *plik;
  int wyn, licz = 0;
  char c;

  plik = fopen(nazwa, "rt");
  printf("Deskryptor: %p\n", plik);

  if(!plik)
  {
    printf("Blad otwierania: %s.\n", nazwa);
    exit(EXIT_FAILURE);
  }

  while((c = fgetc(plik)) != EOF)
  {
    if(c == z)
      licz++;
  }

  wyn = fclose(plik);

  if(wyn)
  {
    printf("Blad zamykania: %s.\n", nazwa);
    exit(EXIT_FAILURE);
  }
  return licz;
}

int main()
{
  char nazwa[] = "plik.txt";
  int l;

  l = wyst(nazwa, 'a');
```

```
  printf("Liczba l: %d", l);

  printf("\n");
  return 0;
}
```

Listing 57. Rozwiązanie zadania nr 64

Zadanie nr 65.

```
#include <stdio.h>
#include <stdlib.h>

int porow(char * nazwa1, char * nazwa2)
{
  FILE *plik1, *plik2;
  char c1 = '\0', c2 = '\0';
  int wyn1, wyn2, p = 1;

  plik1 = fopen(nazwa1, "rt");
  printf("Deskryptor 1: %p\n", plik1);

  if(!plik1)
  {
    printf("Blad otwierania: %s.\n", nazwa1);
    exit(EXIT_FAILURE);
  }

  plik2 = fopen(nazwa2, "rt");
  printf("Deskryptor 2: %p\n", plik2);

  if(!plik2)
  {
    printf("Blad otwierania: %s.\n", nazwa2);
    exit(EXIT_FAILURE);
  }

  while((!feof(plik1)) && (!feof(plik2)))
  {
    c1 = fgetc(plik1);
    c2 = fgetc(plik2);

    if(c1 != c2)
    {
      p = 0;
      break;
    }
  }
```

```c
  wyn1 = fclose(plik1);

  if(wyn1)
  {
    printf("Blad zamykania: %s.\n", nazwa1);
    exit(EXIT_FAILURE);
  }

  wyn2 = fclose(plik2);

  if(wyn2)
  {
    printf("Blad zamykania: %s.\n", nazwa2);
    exit(EXIT_FAILURE);
  }

  return p;
}

int main()
{
  char nazwa1[] = "plik2.txt";
  char nazwa2[] = "plik1.txt";
  int l;

  l = porow(nazwa1, nazwa2);

  printf("Liczba l: %d", l);

  printf("\n");
  return 0;
}
```

Listing 58. Rozwiązanie zadania nr 65

Zadanie nr 66.

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

int porow(char * nazwa1, char * nazwa2)
{
  FILE *plik1, *plik2;
  char c1 = '\0', c2 = '\0';
  int wyn1, wyn2, p = 1;

  plik1 = fopen(nazwa1, "rt");
  printf("Deskryptor 1: %p\n", plik1);
```

```
  if(!plik1)
  {
    printf("Blad otwierania: %s.\n", nazwa1);
    exit(EXIT_FAILURE);
  }

  plik2 = fopen(nazwa2, "rt");
  printf("Deskryptor 2: %p\n", plik2);

  if(!plik2)
  {
    printf("Blad otwierania: %s.\n", nazwa2);
    exit(EXIT_FAILURE);
  }

  while((!feof(plik1)) && (!feof(plik2)))
  {
    while(((c1 = fgetc(plik1)) != EOF) && isspace(c1));
    while(((c2 = fgetc(plik2)) != EOF) && isspace(c2));

    if(c1 != c2)
    {
      p = 0;
      break;
    }
  }

  wyn1 = fclose(plik1);

  if(wyn1)
  {
    printf("Blad zamykania: %s.\n", nazwa1);
    exit(EXIT_FAILURE);
  }

  wyn2 = fclose(plik2);

  if(wyn2)
  {
    printf("Blad zamykania: %s.\n", nazwa2);
    exit(EXIT_FAILURE);
  }

  return p;
}

int main()
{
  char nazwa1[] = "plik.txt";
  char nazwa2[] = "plik2.txt";
```

```
  int l;

  l = porow(nazwa1, nazwa2);

  printf("Liczba l: %d", l);

  printf("\n");
  return 0;
}
```

Listing 59. Rozwiązanie zadania nr 66

Zadanie nr 67.

```
#include <stdio.h>
#include <stdlib.h>

int przepisz(FILE * plik1, FILE * plik2)
{
  char c;
  int p = 0;

  while((c = fgetc(plik1)) != EOF)
  {
    p++;
    fputc(c, plik2);
putchar(c);
  }

  return p;
}

int main()
{
  char nazwa1[] = "plik.txt";
  char nazwa2[] = "plik3.txt";

  FILE *plik1, *plik2;
  int wyn1, wyn2, l;

  plik1 = fopen(nazwa1, "rt");
  printf("Deskryptor 1: %p\n", plik1);

  if(!plik1)
  {
    printf("Blad otwierania: %s.\n", nazwa1);
    exit(EXIT_FAILURE);
  }

  plik2 = fopen(nazwa2, "wt");
```

```c
  printf("Deskryptor 2: %p\n", plik2);

  if(!plik2)
  {
    printf("Blad otwierania: %s.\n", nazwa2);
    exit(EXIT_FAILURE);
  }

  l = przepisz(plik1, plik2);

  printf("Skopiowanych znakow l: %d", l);

  wyn1 = fclose(plik1);

  if(wyn1)
  {
    printf("Blad zamykania: %s.\n", nazwa1);
    exit(EXIT_FAILURE);
  }

  wyn2 = fclose(plik2);

  if(wyn2)
  {
    printf("Blad zamykania: %s.\n", nazwa2);
    exit(EXIT_FAILURE);
  }

  printf("\n");
  return 0;
}
```

Listing 60. Rozwiązanie zadania nr 67

Zadanie nr 68.

```c
#include <stdio.h>
#include <stdlib.h>

void zapisz(char * nazwa, int ** tab, int n, int m)
{
  FILE *plik;
  int i, wyn;

  plik = fopen(nazwa, "wb");
  printf("Deskryptor: %p\n", plik);

  if(!plik)
  {
    printf("Blad otwierania: %s.\n", nazwa);
```

```
    exit(EXIT_FAILURE);
  }

  for(i = 0; i < n; i++)
    fwrite(tab[i], sizeof(int), m, plik);

  wyn = fclose(plik);

  if(wyn)
  {
    printf("Blad zamykania: %s.\n", nazwa);
    exit(EXIT_FAILURE);
  }
}

int main()
{
  char nazwa[] = "tab.bin";
  int n = 5, m = 5, i, j, **t;

  t = malloc(n * sizeof(int *));

  for(i = 0; i < n; i++)
    t[i] = malloc(m * sizeof(int));

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      t[i][j] = i + j;

  printf("Wartosci w t:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  zapisz(nazwa, t, n, m);

  for(int i = 0; i < n; i++)
    free(t[i]);
  free(t);

  printf("\n");
  return 0;
}
```

Listing 61. Rozwiązanie zadania nr 68

Zadanie nr 69.

```c
#include <stdio.h>
#include <stdlib.h>

void wczytaj(char * nazwa, int ** tab, int n, int m)
{
  FILE *plik;
  int i, wyn;

  plik = fopen(nazwa, "rb");
  printf("Deskryptor: %p\n", plik);

  if(!plik)
  {
    printf("Blad otwierania: %s.\n", nazwa);
    exit(EXIT_FAILURE);
  }

  for(i = 0; i < n; i++)
    fread(tab[i], sizeof(int), m, plik);

  wyn = fclose(plik);

  if(wyn)
  {
    printf("Blad zamykania: %s.\n", nazwa);
    exit(EXIT_FAILURE);
  }
}

int main()
{
  char nazwa[] = "tab.bin";
  int n = 5, m = 5, i, j, **t;

  t = malloc(n * sizeof(int *));

  for(i = 0; i < n; i++)
    t[i] = malloc(m * sizeof(int));

  printf("Wartosci w t:\n");

  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  wczytaj(nazwa, t, n, m);

  printf("Wartosci w t:\n");
```

```
  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      printf("[%d,%d]=%d%s", i, j, t[i][j], (j < m - 1)?", ":"\n");

  for(int i = 0; i < n; i++)
    free(t[i]);
  free(t);

  printf("\n");
  return 0;
}
```

Listing 62. Rozwiązanie zadania nr 69

9. Literatura

1. Prata S. (2016). Język C. Szkoła programowania. Wydawnictwo Helion. Gliwice.
2. Koczubiej S. (2025). Podstawy programowania. (https://staff.tu.kielce.pl/sk/media/downloads/pp/pp-wyklad.pdf, dostępny 03.02.2025).
3. C reference. (n.d.). (https://en.cppreference.com/w/c dostępny 03.02.2025).
4. What is MSYS2? (n.d.). (https://www.msys2.org/docs/what-is-msys2, dostępny 03.02.2025).

10. Spis rysunków

11. Spis listingów