C/C++ dla Visual Studio Code

Instalacja MinGW-x64 na Windows

1. Pobierz instalator MSYS2 ze strony https://www.msys2.org/

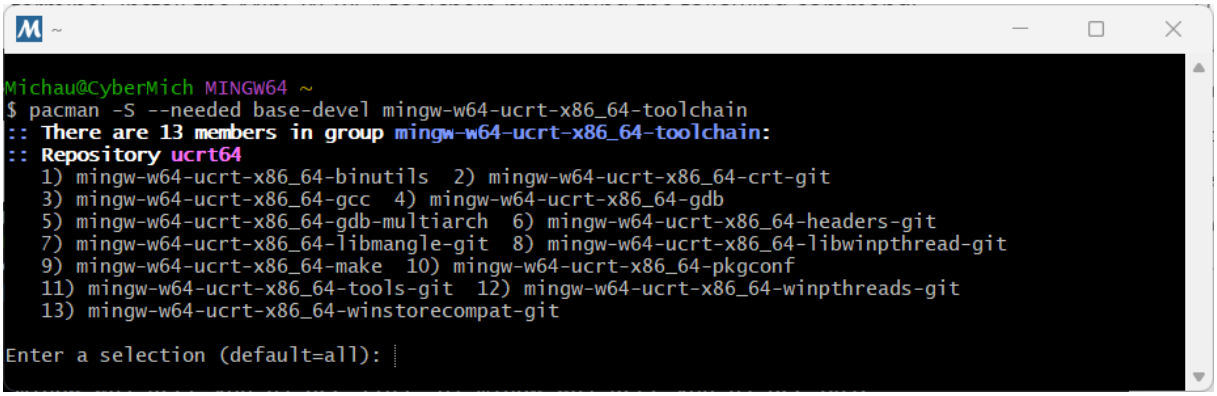
2. Uruchom instalator i postępuj zgodnie z instrukcjami kreatora instalacji. Uwaga, MSYS2 wymaga 64-bitowego systemu Windows 8.1 lub nowszego.

3. W kreatorze wybierz żądany folder instalacyjny. Wpisz folder do instalacji: tylko lokalny folder, bez spacji, polskich znaków etc. Najlepiej zostaw domyślny **C:\msys64**

4. W większości przypadków zalecany katalog jest akceptowalny. To samo dotyczy kroku ustawiania skrótów menu startowego. Po zakończeniu upewnij się, że pole wyboru Uruchom **MSYS2MINGW64** teraz jest zaznaczone i wybierz Zakończ. Następnie automatycznie otworzy się okno terminala MSYS2.

5. W tym terminalu zainstaluj łańcuch narzędzi MinGW-w64, uruchamiając następujące polecenie:

   pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain

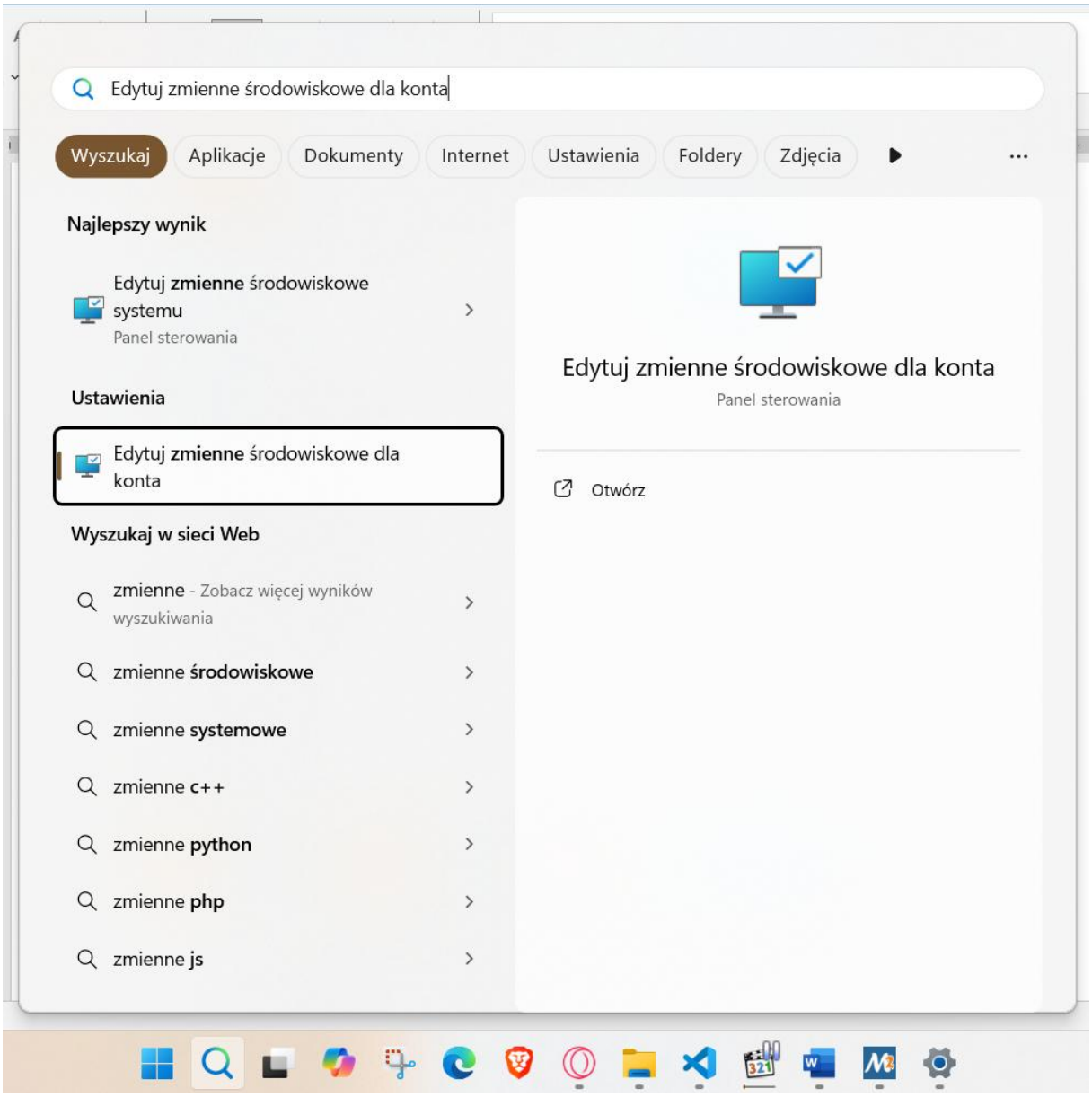


6. Zaakceptuj domyślną liczbę pakietów w grupie narzędzi, naciskając Enter.
7. Wpisz Y, gdy pojawi się pytanie, czy kontynuować instalację.

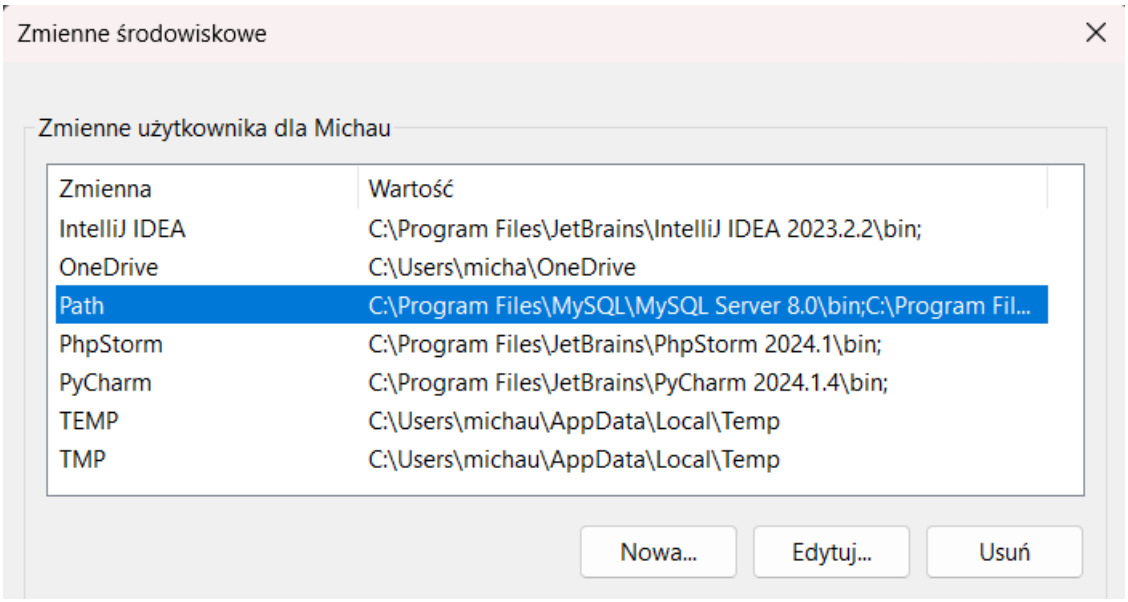
Dodanie zmiennej środowiskowej PATH systemu Windows

1. Dodaj ścieżkę folderu bin MinGW-w64 do zmiennej środowiskowej PATH systemu Windows, wykonując następujące kroki:

a) W pasku wyszukiwania systemu Windows wpisz słowo: „zmienne”, aby wyszukać „Edytuj zmienne środowiskowe dla konta”

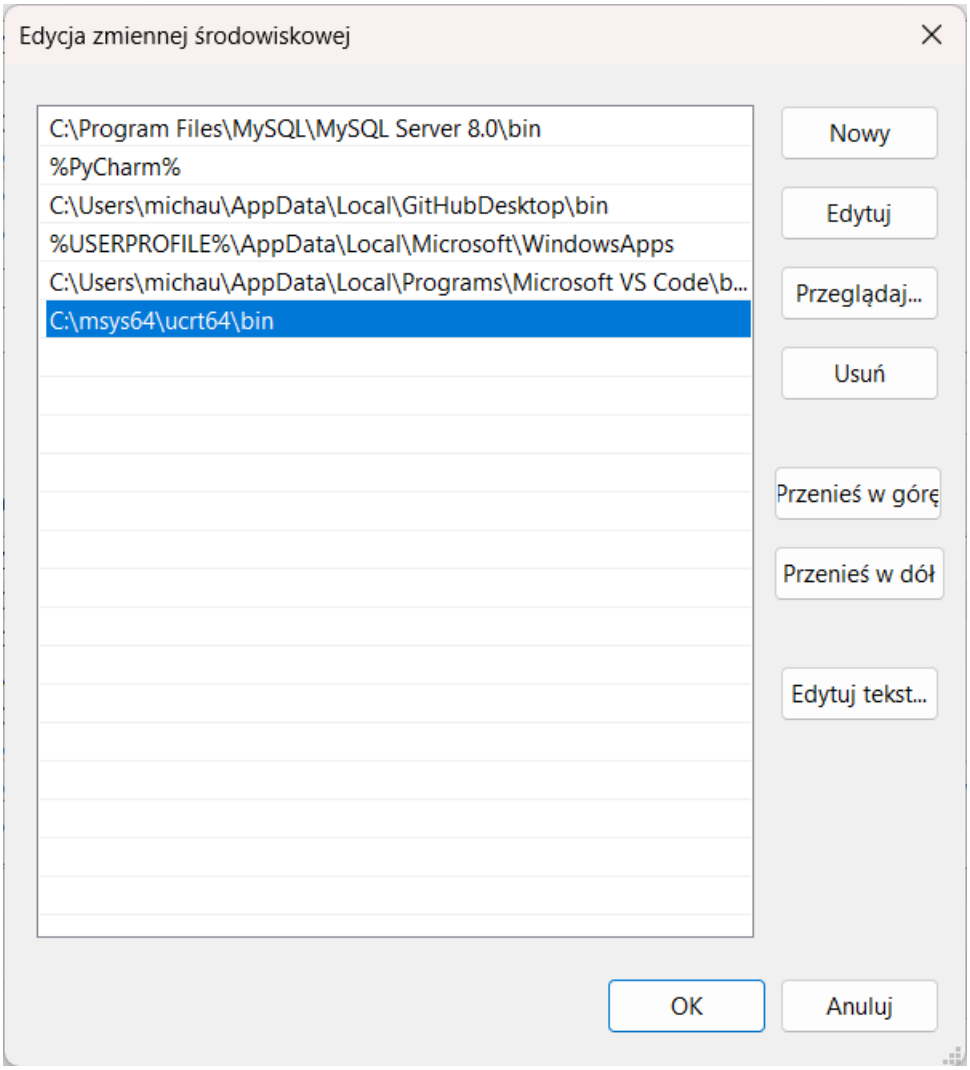


b) W zmiennych użytkownika wybierz zmienną **Path**, a następnie wybierz **Edytuj**.



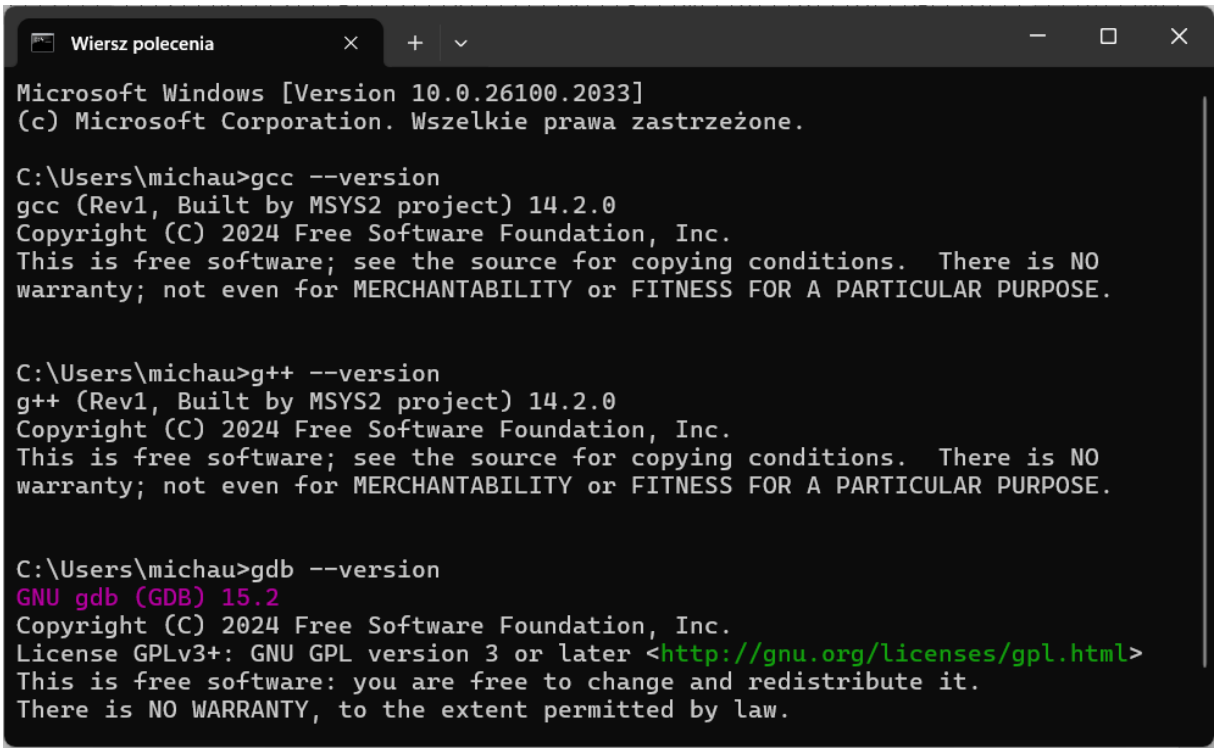
c) Wybierz **Nowy** i dodaj do listy folder docelowy MinGW-w64, który zapisałeś podczas procesu instalacji. Jeśli wybrałeś domyślne kroki instalacji, ścieżka to:

C:\msys64\ucrt64\bin



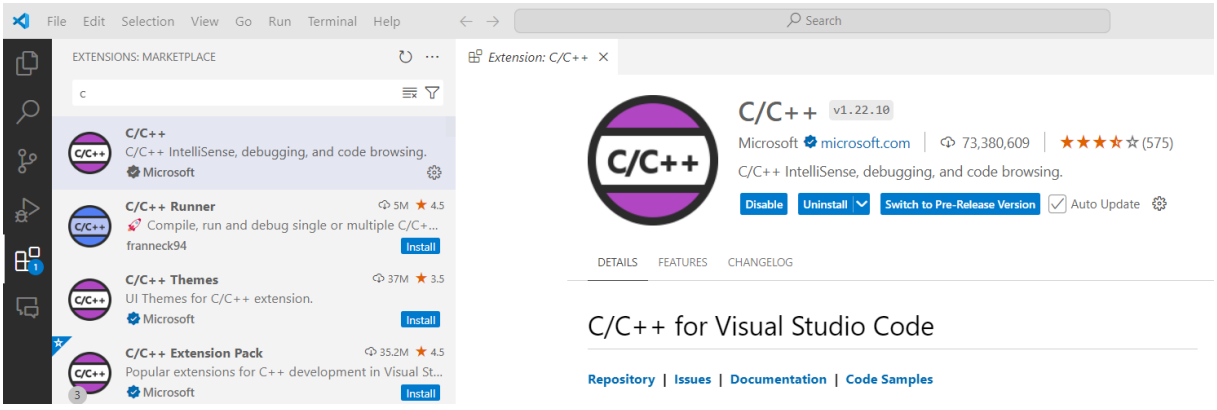
d) Wybierz **OK**, a następnie ponownie wybierz **OK** w oknie Zmienne środowiskowe, aby zaktualizować zmienną środowiskową PATH.
e) Otwórz ponownie konsolę i sprawdź polecenia:

gcc –version
g++ --version
gdb –version



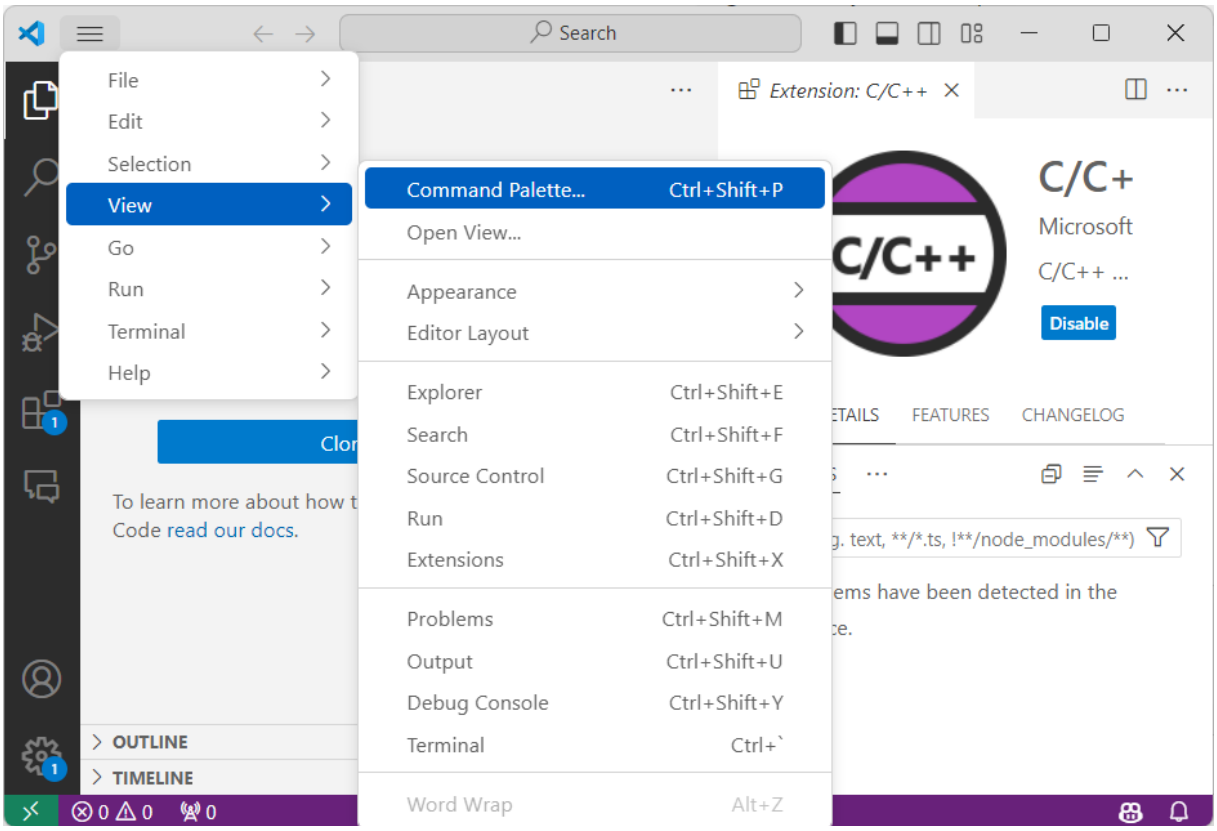
Zainstaluj rozszerzenie C/C++ for Visual Studio Code

1. Otwórz **Visual Studio Code**.
2. Wybierz ikonę widoku **Extensions** na pasku aktywności lub użyj skrótu klawiaturowego (Ctrl+Shift+X).
3. Wyszukaj **C/C++**.
4. Wybierz **Install**.

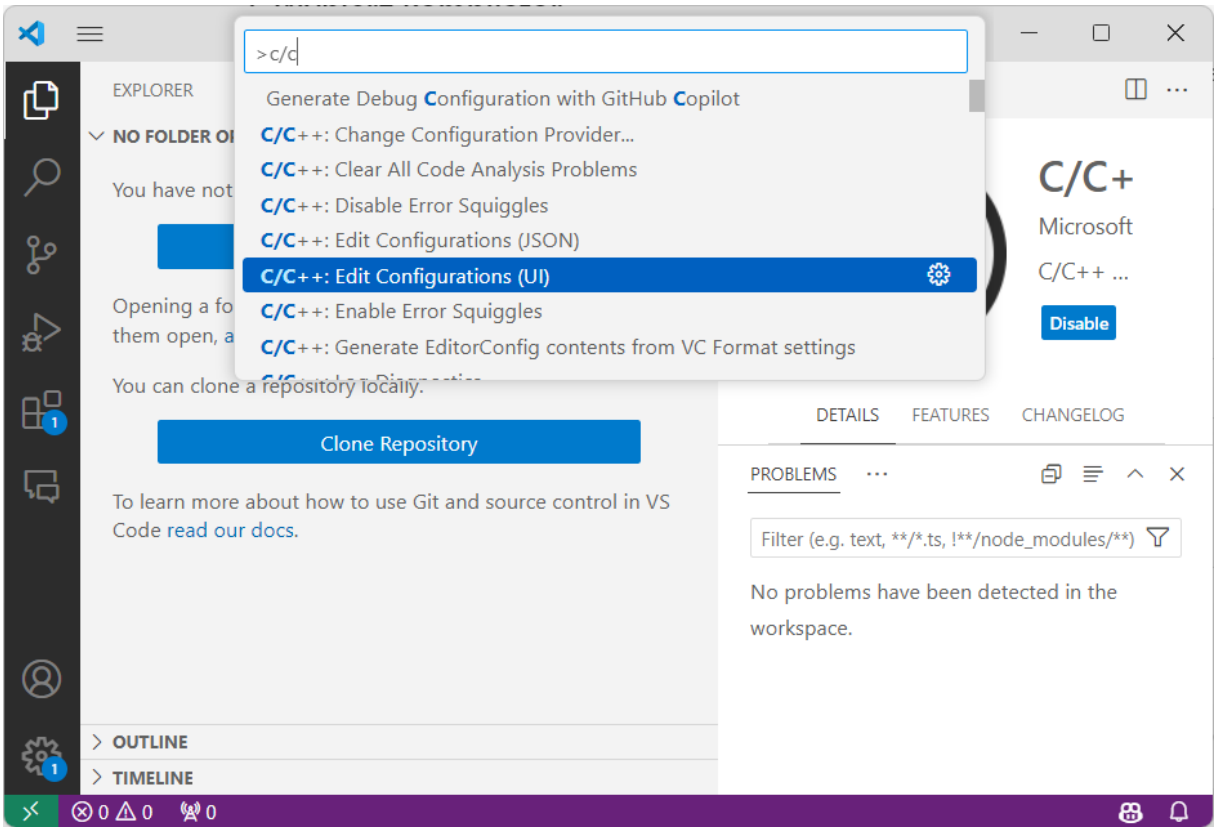


Kolejne kroki

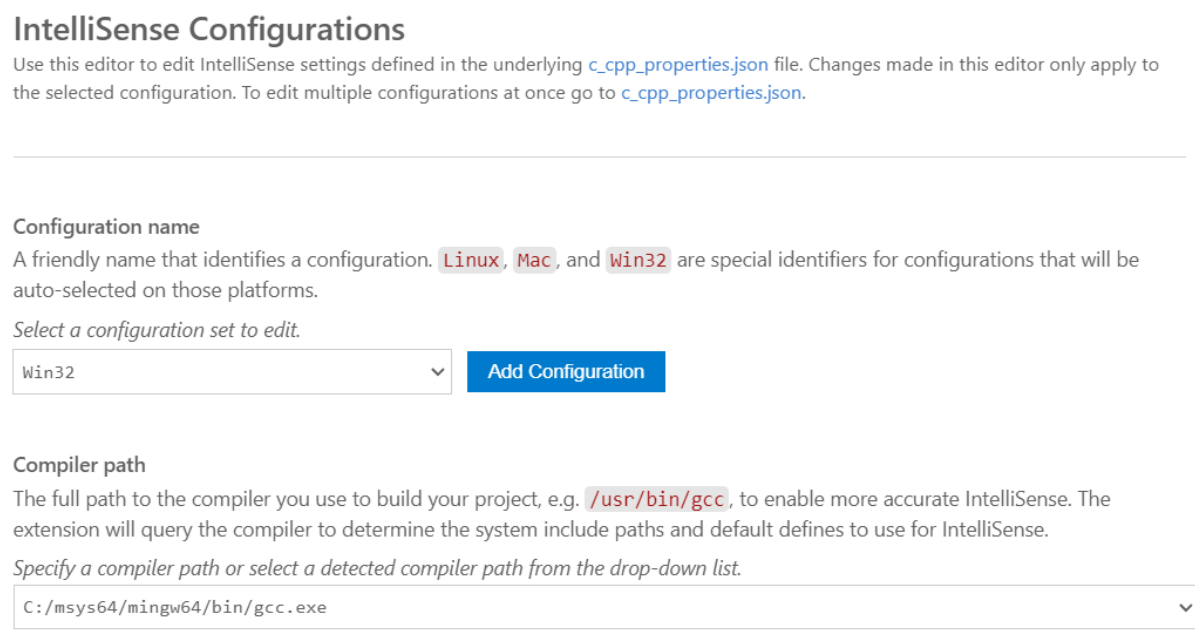
1. Opcja **View** \ **Command Palette** (Ctr+Shift+P)



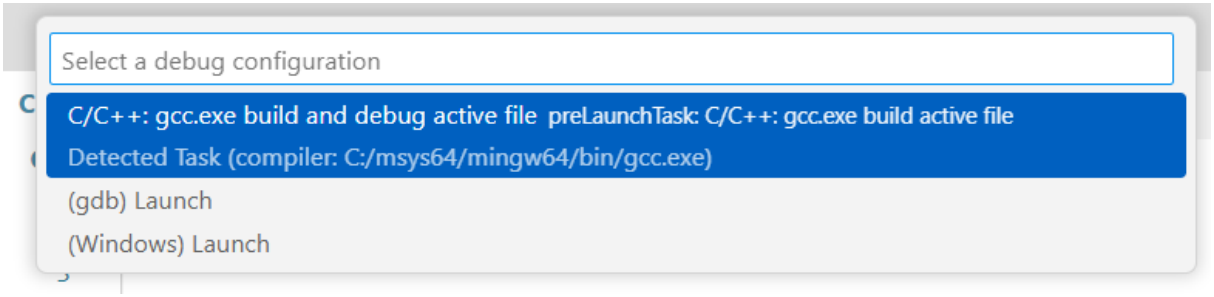
Wyszukaj **C/C++:Edit Configuration (UI)**



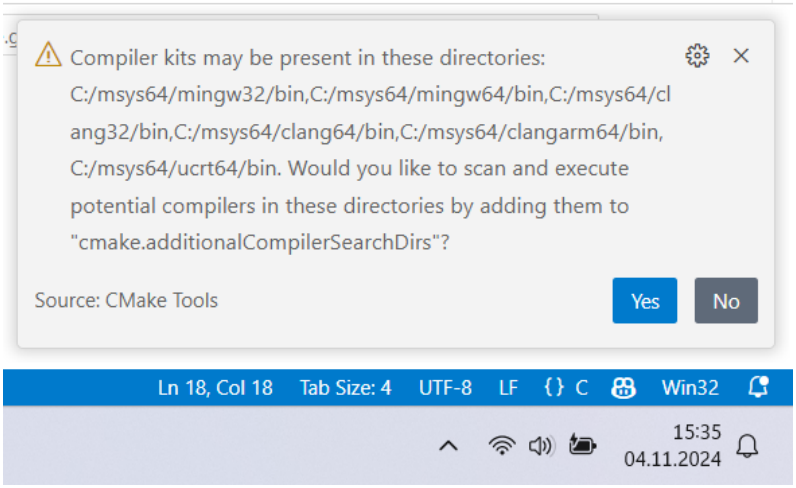
Wybierz w opcjach Compiler path lokalizację **C:\msys64\mingw64\bin\gcc.exe**



Przy kompilacji wybierz tę samą ścieżkę



Zatwierdź



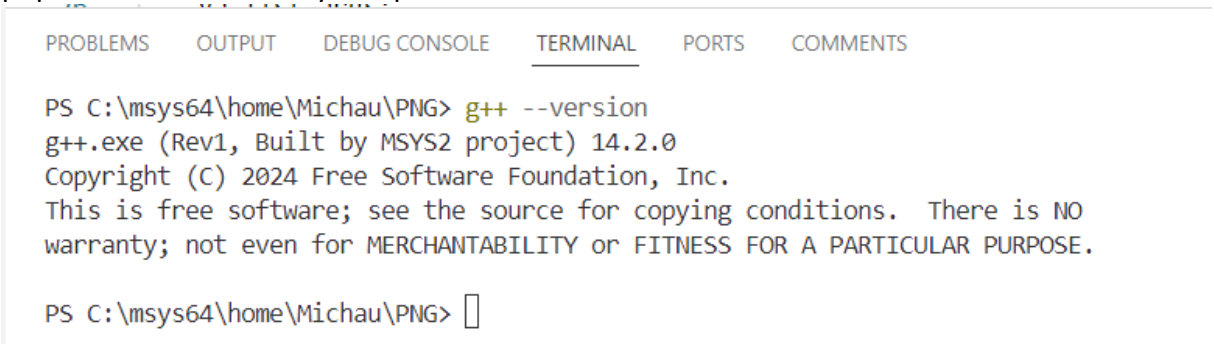
Skonfiguruj swoje środowisko C/C++

Sprawdź, czy masz zainstalowany kompilator:

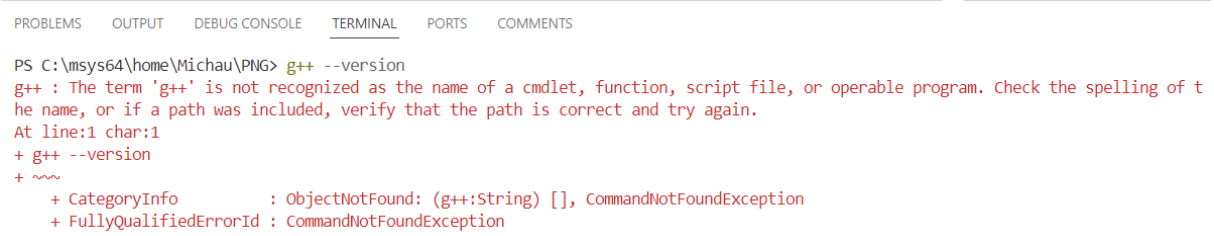
1. Otwórz nowe okno terminala VS Code za pomocą (**Ctrl+Shift+`**)
2. Użyj następującego polecenia, aby sprawdzić kompilator GCC g++:

g++ --version

poprawnie zainstalowany kompilator:



I przykład nie znalezionego kompilatora:



Wynik powinien pokazać wersję kompilatora i szczegóły. Jeśli nie znaleziono żadnej z nich, upewnij się, że plik wykonywalny kompilatora znajduje się w odpowiedniej ścieżce (%PATH w systemie Windows, $PATH w systemie Linux i macOS), aby rozszerzenie C/C++ mogło go znaleźć.

Dodanie blibioteki-libpng

Aby dodać bibliotekę `libpng` do obecnej konfiguracji w pliku tasks.json, musisz dodać flagę `-lpng` do argumentów kompilatora. Oto jak możesz to zrobić:

W domyślnym folderze projektu:

C:\msys64\home\Nazwa_użytkownika\PNG\.vscode\tasks.json

W pliku **task.json** dopisać w podanym miejscu linijkę:

„-lpng”

```
"args": [
    "-fdiagnostics-color=always",
    "-g",
    "${file}",
    "-o",
    "${fileDirname}\\${fileBasenameNoExtension}.exe",
    "-lpng"
],
```

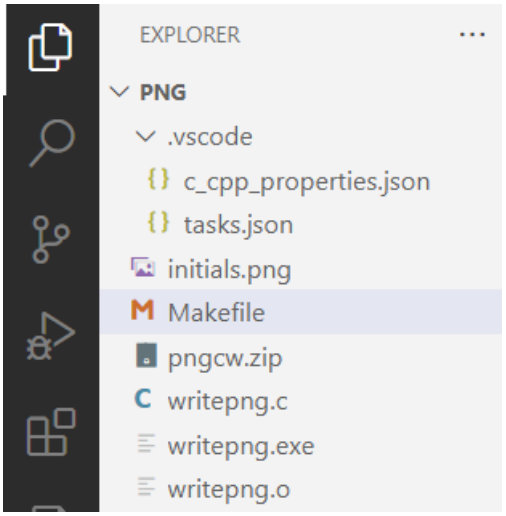
Podczas uruchamiania programu wszystko działa poprawnie:

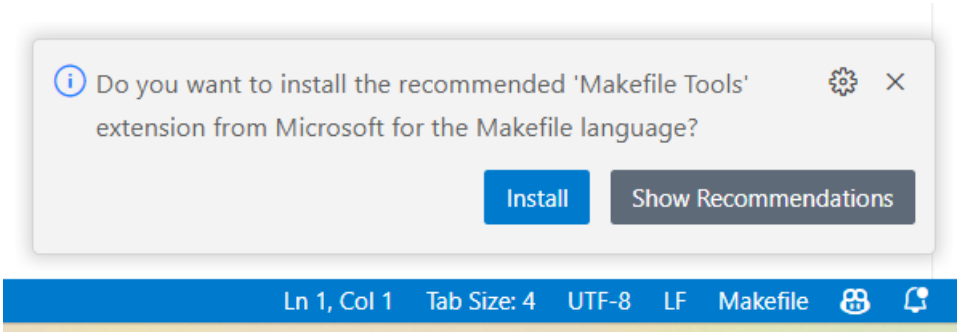
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

Loaded 'C:\WINDOWS\SYSTEM32\cryptbase.dll'. Symbols loaded.
Loaded 'C:\WINDOWS\System32\bcryptprimitives.dll'. Symbols loaded.
[Thread 12832.0x3d70 exited with code 0]
[Thread 12832.0x1fe4 exited with code 0]
[Thread 12832.0x438c exited with code 0]
[Inferior 1 (process 12832) exited normally]
The program 'C:\msys64\home\Michau\PNG\writepng.exe' has exited with code 0 (0x00000000).
```

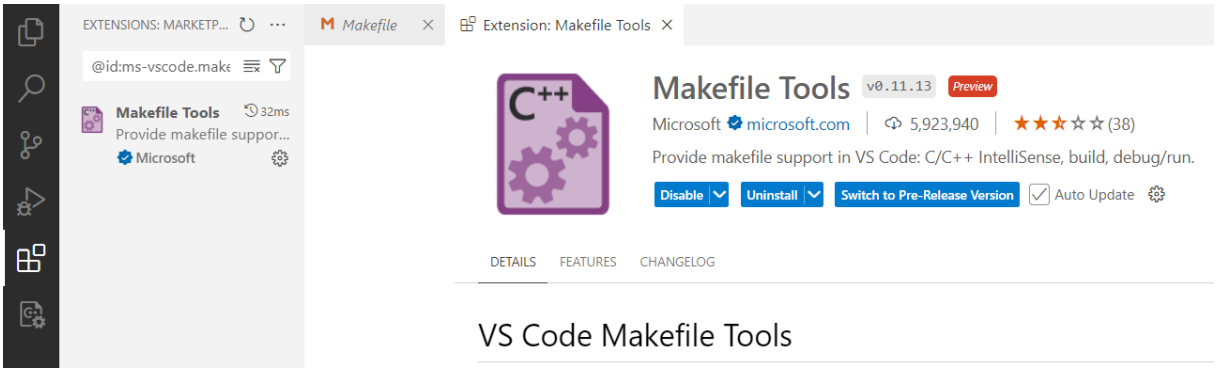
Dodanie rozszerzenia Makefile Tools

Po kliknięciu na plik projektu Makefile dostaniesz komunikat





Kliknij **Install**



Aby uruchomić i debugować program `writepng` w Visual Studio Code, trzeba skonfigurować kilka plików.

Konfiguracja kompilatora w katalogu .vscode

c_cpp_properties.json

- Upewnij się, że ścieżka do kompilatora GCC jest poprawnie ustawiona:

  "compilerPath": "C:/msys64/mingw64/bin/gcc.exe"

Konfiguracja zadania kompilacji w pliku

tasks.json

Dodaj zadanie kompilacji, które używa GCC do kompilacji aktywnego pliku:

writepng.c | launch.json | c_cpp_properties.json | tasks.json

.vscode > tasks.json > ...

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: gcc.exe build active file",
            "command": "C:/msys64/mingw64/bin/gcc.exe",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe",
                "-lpng"
            ],
            "options": {
                "cwd": "C:/msys64/mingw64/bin",
                "env": {
                    "PATH": "${env:PATH};C:/msys64/mingw64/bin"
                }
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "Task generated by Debugger."
        }
    ]
}
```

Konfiguracja uruchamiania i debugowania w pliku

launch.json

Dodaj konfigurację debugowania, która uruchamia program writepng.exe i używa GDB jako debugera:

writepng.c | launch.json | c_cpp_properties.json | tasks.json

.vscode > launch.json > ...

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Run writepng",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/writepng.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "miDebuggerPath": "C:/msys64/mingw64/bin/gdb.exe",
            "setupCommands": [
                {
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ],
            "preLaunchTask": "C/C++: gcc.exe build active file",
            "internalConsoleOptions": "openOnSessionStart"
        }
    ]
}
```

Makefile

Upewnij się, że Makefile jest poprawnie skonfigurowany do kompilacji programu `writepng`:

writepng.c launch.json *Makefile* tasks.json

Makefile

```
CC = gcc
SOURCES = $(wildcard  *.c)
OBJECTS = $(SOURCES:.c=.o)
TARGET = initials.png
CFLAGS = -g -Wall

ifeq ($(OS),Windows_NT)
LIBS = -l:libpng.a -l:libz.a -lm
EXECUTABLE = writepng.exe
else
LIBS = -lpng -lm
EXECUTABLE = writepng
endif


all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
    $(CC) $(OBJECTS) -o $@ $(LIBS)

clean:
    rm -f $(EXECUTABLE) $(OBJECTS) $(TARGET)

run: $(EXECUTABLE)
    ./$(EXECUTABLE)

$(TARGET): run

test: run
    display $(TARGET)

```

Dodatkowe pliki w katalogu projektu

Dodaj pliki z lokalizacji C:\msys64\mingw64\bin do lokalizacji C:\msys64\home\nazwa_użytkownika\PNG

libpng16-16.dll
zlib1.dll

Kod źródłowy

writepng.c

Upewnij się, że kod źródłowy jest poprawnie napisany i zawiera wszystkie niezbędne funkcje do tworzenia i zapisywania pliku PNG.

Dzięki tym zmianom, możesz teraz kompilować, uruchamiać i debugować program `writepng` bezpośrednio w Visual Studio Code.