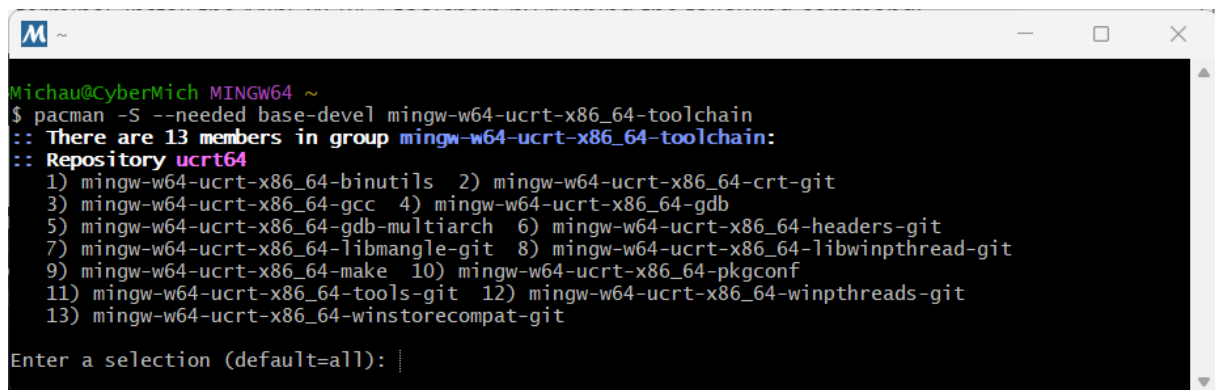


# C/C++ dla Visual Studio Code

## 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***



```
Michau@CyberMich MINGW64 ~
$ pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain
:: There are 13 members in group mingw-w64-ucrt-x86_64-toolchain:
:: Repository ucrt64
 1) mingw-w64-ucrt-x86_64-binutils  2) mingw-w64-ucrt-x86_64-crt-git
 3) mingw-w64-ucrt-x86_64-gcc      4) mingw-w64-ucrt-x86_64-gdb
 5) mingw-w64-ucrt-x86_64-gdb-multiarch  6) mingw-w64-ucrt-x86_64-headers-git
 7) mingw-w64-ucrt-x86_64-libmangle-git  8) mingw-w64-ucrt-x86_64-libwinpthread-git
 9) mingw-w64-ucrt-x86_64-make    10) mingw-w64-ucrt-x86_64-pkgconf
11) mingw-w64-ucrt-x86_64-tools-git 12) mingw-w64-ucrt-x86_64-winpthreads-git
13) mingw-w64-ucrt-x86_64-winstorecompat-git

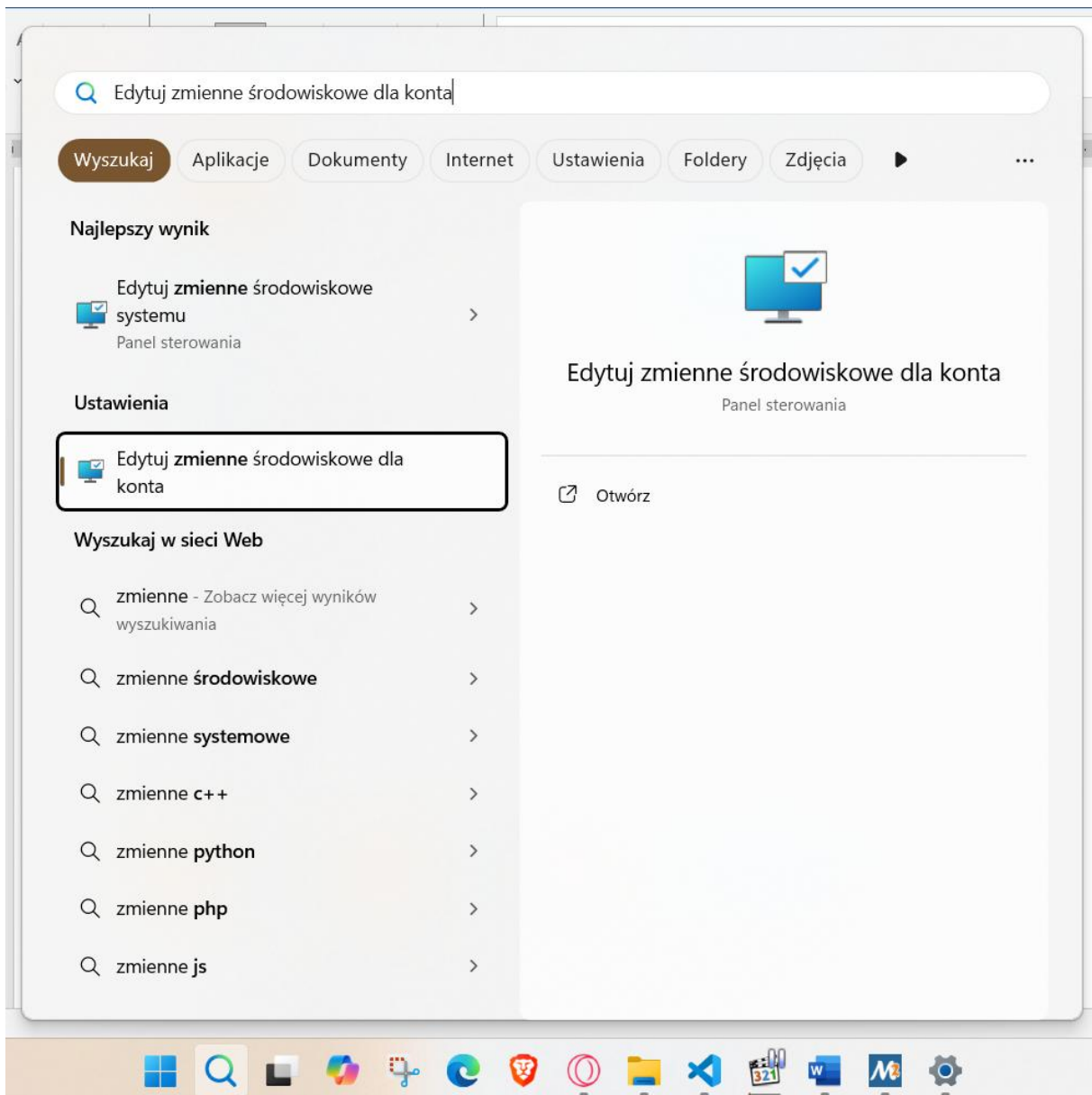
Enter a selection (default=all):
```

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

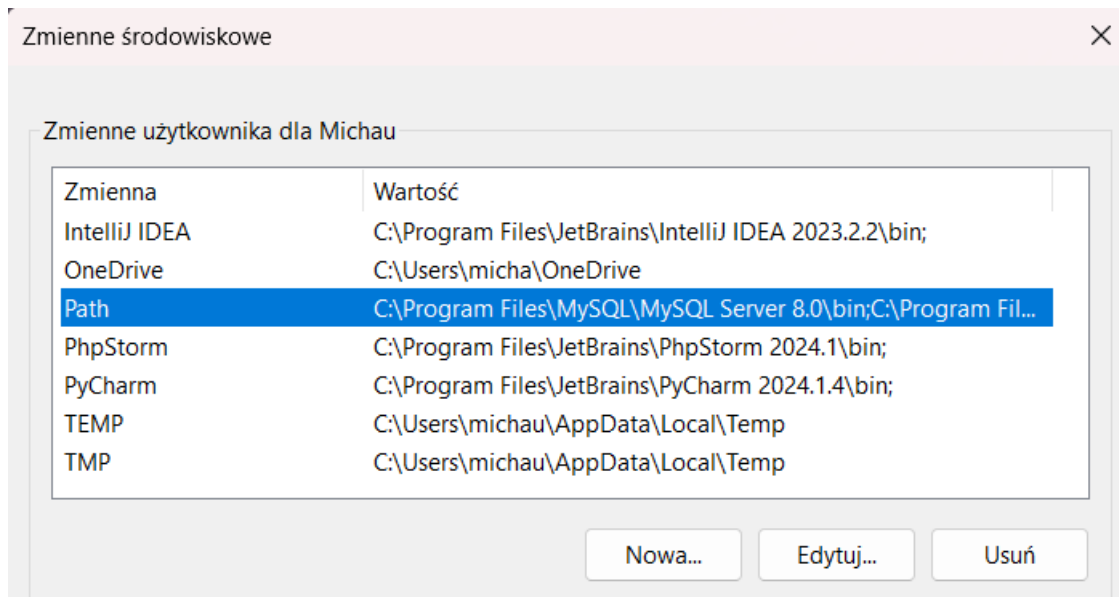
## Dodanie zmiennej środowiskowej PATH systemu Windows

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”

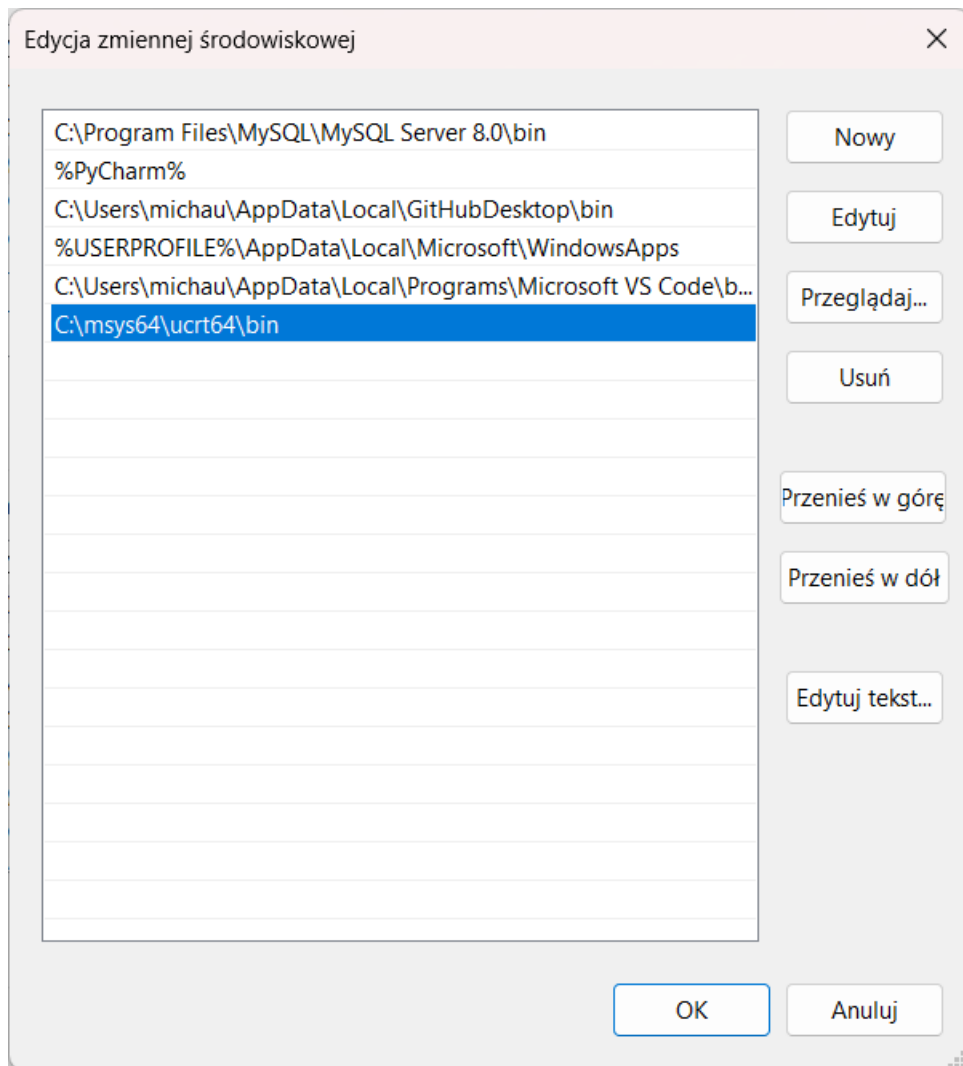


- b) W zmiennych użytkownika wybierz zmienną **Path**, a następnie wybierz **Edytuj**.



- 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***

```
Wiersz polecenia
Microsoft Windows [Version 10.0.26100.2033]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

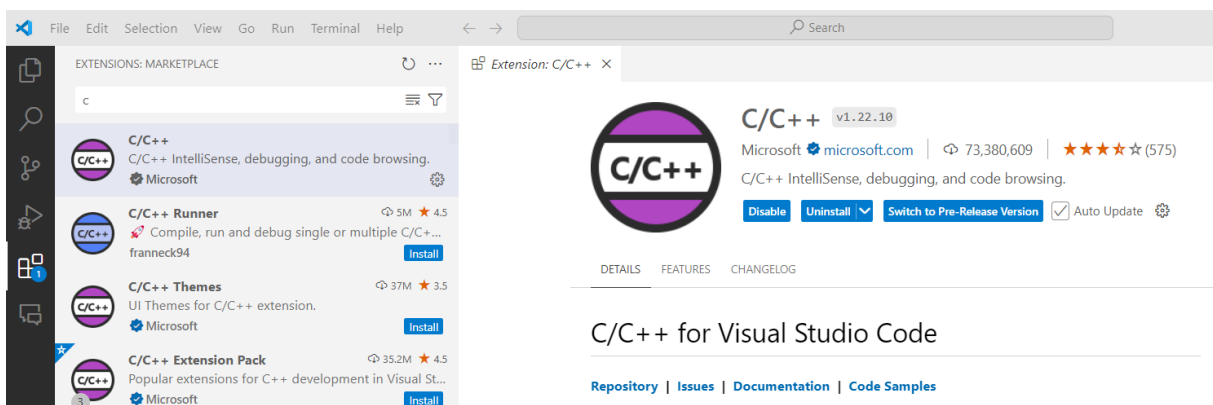
C:\Users\michau>gcc --version
gcc (Rev1, Built by MSYS2 project) 14.2.0
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

C:\Users\michau>g++ --version
g++ (Rev1, Built by MSYS2 project) 14.2.0
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

C:\Users\michau>gdb --version
GNU gdb (GDB) 15.2
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

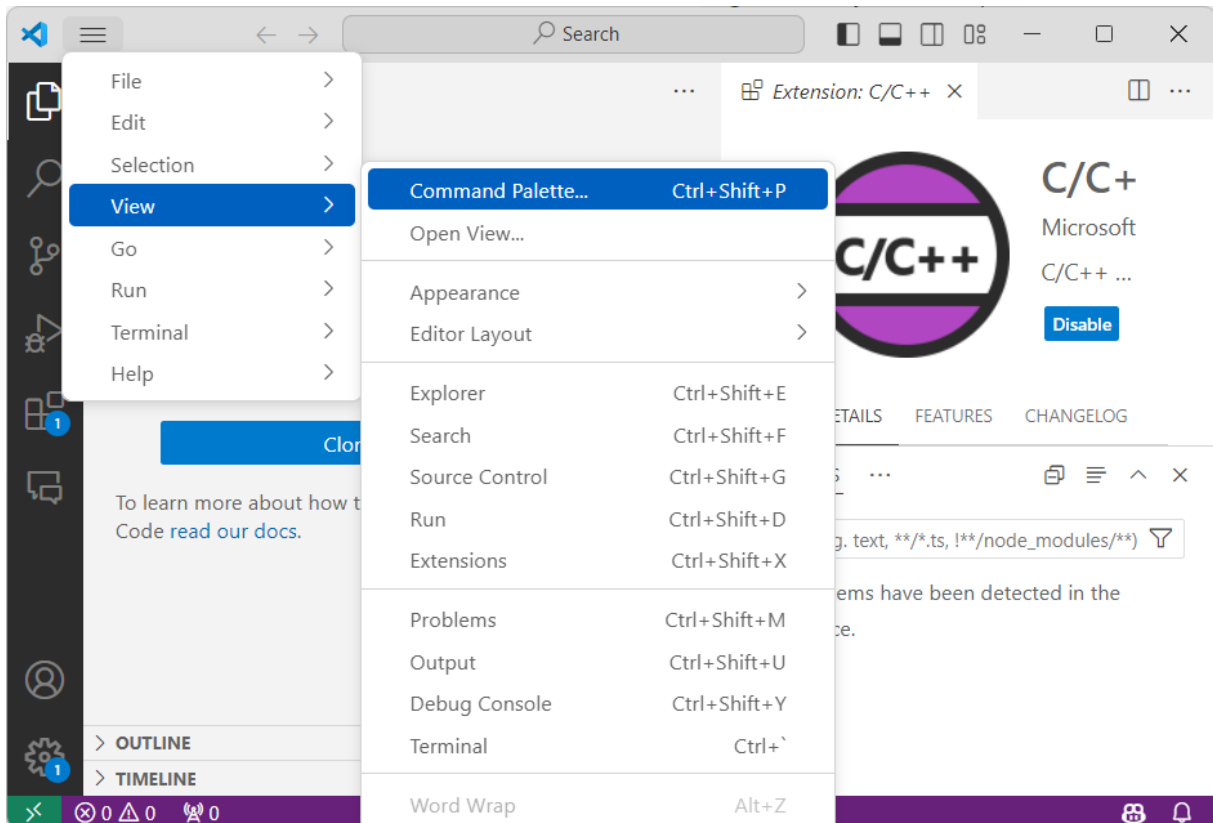
## Zainstaluj rozszerzenie C/C++ for Visual Studio Code

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**.

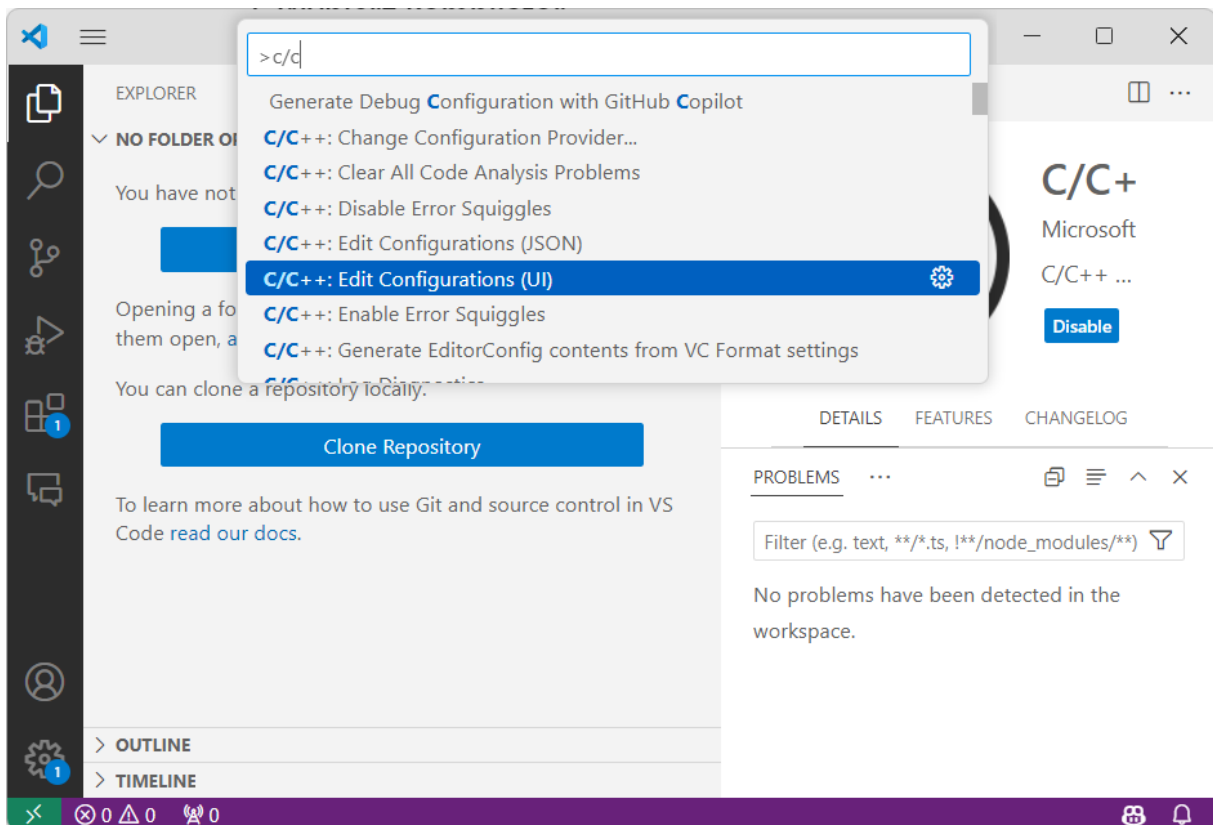


## Kolejne kroki

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



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



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

## IntelliSense Configurations

Use this editor to edit IntelliSense settings defined in the underlying `c_cpp_properties.json` file. Changes made in this editor only apply to the selected configuration. To edit multiple configurations at once go to `c_cpp_properties.json`.

### Configuration name

A friendly name that identifies a configuration. `Linux`, `Mac`, and `Win32` are special identifiers for configurations that will be auto-selected on those platforms.

Select a configuration set to edit.

Win32

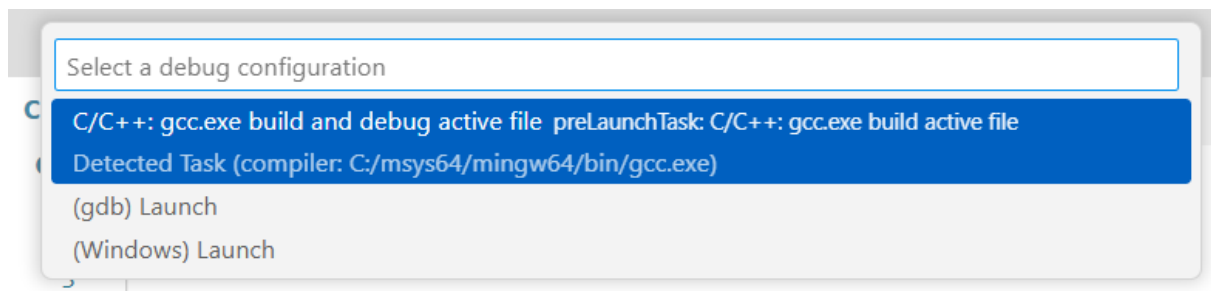
### Compiler path

The full path to the compiler you use to build your project, e.g. `/usr/bin/gcc`, to enable more accurate IntelliSense. The extension will query the compiler to determine the system include paths and default defines to use for IntelliSense.

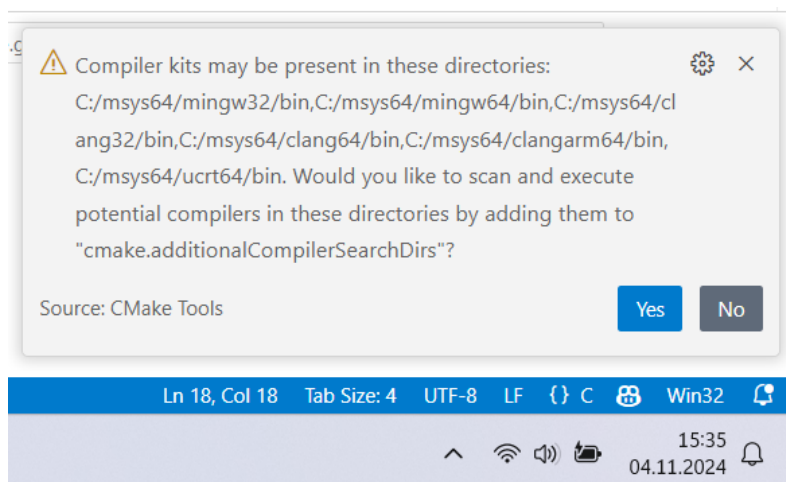
Specify a compiler path or select a detected compiler path from the drop-down list.

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:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

PS C:\msys64\home\Michau\PNG> g++ --version
g++.exe (Rev1, Built by MSYS2 project) 14.2.0
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

PS C:\msys64\home\Michau\PNG> █
```

I przykład nie znalezionego kompilatora:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

PS C:\msys64\home\Michau\PNG> g++ --version
g++ : The term 'g++' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the
name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ g++ --version
+ ~~~
+ CategoryInfo          : ObjectNotFound: (g++:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

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 biblioteki-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”***



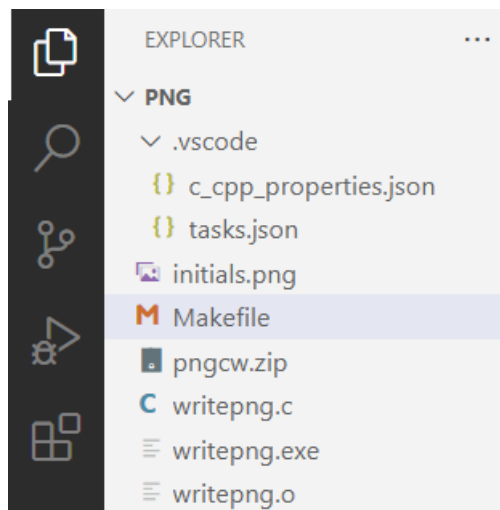
```
"args": [  
  "-fdiagnostics-color=always",  
  "-g",  
  "${file}",  
  "-o",  
  "${fileDirname}\\${fileBasenameNoExtension}.exe",  
  "-lpng"  
],
```

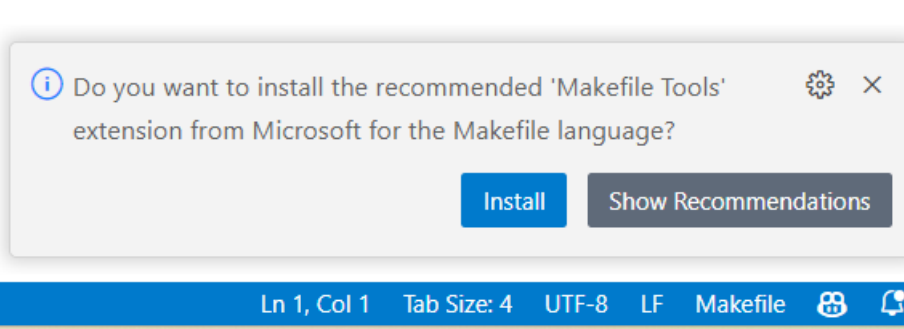
Podczas uruchamiania programu wszystko działa poprawnie:

```
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).
```

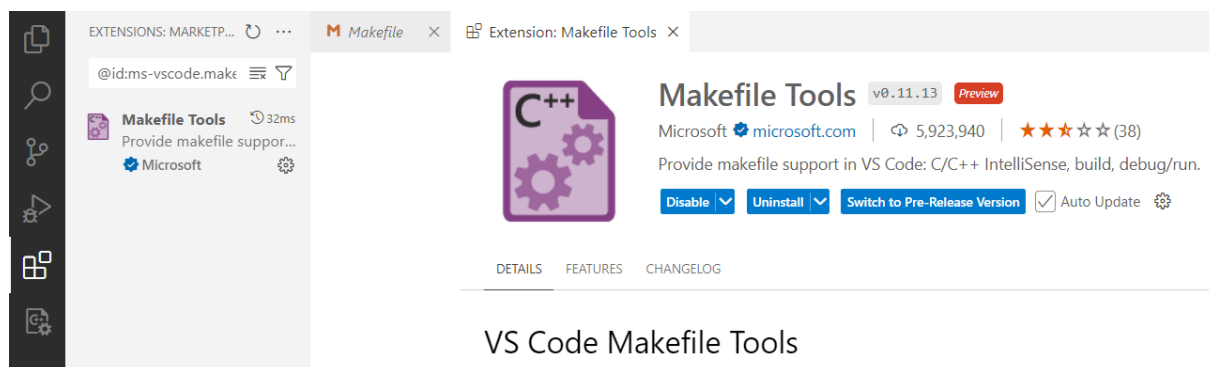
## Dodanie rozszerzenia Makefile Tools

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"
```

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a project named 'PNG' with a file tree including .vscode, c\_cpp\_properties.json, launch.json, tasks.json, initials.png, libpng16-16.dll, Makefile, pngcw.zip, writepng.c, writepng.exe, writepng.o, and zlib1.dll. The main editor window shows the content of c\_cpp\_properties.json, which is a JSON configuration for the C/C++ compiler. The configuration includes a 'name' of 'Win32', an 'includePath' pointing to the workspace folder, 'defines' for '\_DEBUG' and 'UNICODE', and a 'compilerPath' pointing to 'C:/msys64/mingw64/bin/gcc.exe'. The 'version' is set to 4.

```
1 {
2   "configurations": [
3     {
4       "name": "Win32",
5       "includePath": [
6         "${workspaceFolder}/**"
7       ],
8       "defines": [
9         "_DEBUG",
10        "UNICODE",
11        "_UNICODE"
12      ],
13       "compilerPath": "C:/msys64/mingw64/bin/gcc.exe"
14     }
15  ],
16  "version": 4
17 }
```

## Konfiguracja zadania kompilacji w pliku

### tasks.json

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

```
.vscode > {} tasks.json > ...
1  {
2    "version": "2.0.0",
3    "tasks": [
4      {
5        "type": "cppbuild",
6        "label": "C/C++: gcc.exe build active file",
7        "command": "C:/msys64/mingw64/bin/gcc.exe",
8        "args": [
9          "-fdiagnostics-color=always",
10         "-g",
11         "${file}",
12         "-o",
13         "${fileDirname}\\${fileBasenameNoExtension}.exe",
14         "-lpng"
15       ],
16       "options": {
17         "cwd": "C:/msys64/mingw64/bin",
18         "env": {
19           "PATH": "${env:PATH};C:/msys64/mingw64/bin"
20         }
21       },
22       "problemMatcher": [
23         "$gcc"
24       ],
25       "group": {
26         "kind": "build",
27         "isDefault": true
28       },
29       "detail": "Task generated by Debugger."
30     }
31   ]
32 }
```

## Konfiguracja uruchamiania i debugowania w pliku

### launch.json

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

```
C writepng.c  {} launch.json ×  {} c_cpp_properties.json  {} tasks.json
.vscode > {} launch.json > ...
1  {
2  // Use IntelliSense to learn about possible attributes.
3  // Hover to view descriptions of existing attributes.
4  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5  "version": "0.2.0",
6  "configurations": [
7    {
8      "name": "Run writepng",
9      "type": "cppdbg",
10     "request": "launch",
11     "program": "${workspaceFolder}/writepng.exe",
12     "args": [],
13     "stopAtEntry": false,
14     "cwd": "${workspaceFolder}",
15     "environment": [],
16     "externalConsole": false,
17     "MIMode": "gdb",
18     "miDebuggerPath": "C:/msys64/mingw64/bin/gdb.exe",
19     "setupCommands": [
20       {
21         "description": "Enable pretty-printing for gdb",
22         "text": "-enable-pretty-printing",
23         "ignoreFailures": true
24       }
25     ],
26     "preLaunchTask": "C/C++: gcc.exe build active file",
27     "internalConsoleOptions": "openOnSessionStart"
28   }
29 ]
30 }
```

## Makefile

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

```
C writepng.c × {} launch.json M Makefile × {} tasks.json
M Makefile
1 CC = gcc
2 SOURCES = $(wildcard *.c)
3 OBJECTS = $(SOURCES:.c=.o)
4 TARGET = initials.png
5 CFLAGS = -g -Wall
6
7 ifeq ($(OS),Windows_NT)
8 LIBS = -l:libpng.a -l:libz.a -lm
9 EXECUTABLE = writepng.exe
10 else
11 LIBS = -lpng -lm
12 EXECUTABLE = writepng
13 endif
14
15
16 all: $(SOURCES) $(EXECUTABLE)
17
18 $(EXECUTABLE): $(OBJECTS)
19 | $(CC) $(OBJECTS) -o $@ $(LIBS)
20 |
21 clean:
22 | rm -f $(EXECUTABLE) $(OBJECTS) $(TARGET)
23 |
24 run: $(EXECUTABLE)
25 | ./$(EXECUTABLE)
26 |
27 $(TARGET): run
28
29 test: run
30 | display $(TARGET)
31
```

## 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.