

Narzędzia programistyczne

- **kompilator** + edytor tekstu
- **debugger** (odpluskwiacz) - służy do dynamicznej analizy programów (w czasie działania), w celu odnalezienia i identyfikacji zawartych w nich błędów, np. śledzenie wartości zmiennych, stan stosu
- **profilowanie** - badanie zachowania programu, przy użyciu informacji zdobytych podczas jego wykonywania np. graf wywołań, pomiar czasu wykonywania instrukcji, badanie zajętości pamięci
- **analiza statyczna kodu (linter)** - analiza fragmentów kodu (bez konieczności uruchamiania) w celu wykrycia potencjalnych błędów, naruszeń stylu (tzw. *code smells*)
- **systemy kontroli wersji** - śledzenie zmian w kodzie i współdzielenie kodu w zespołach, scalanie zmian
- środowiska IDE - integrują wiele narzędzi

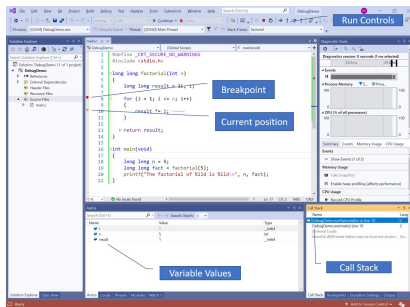
Przydatne opcje kompilatora GCC:

-Wall	włącza szereg reguł wykrywających potencjalne błędy, ge
-Wextra	włącza dodatkowe reguły dotyczące ostrzeżeń
-ansi	używa standardu ANSI C89
-pedantic	pilnuje restrykcyjnie norm standardu ISO
-fanalyzer	analiza statyczna kodu
-g	dodaje do wynikowego kodu informacje niezbędne do deb
-pb	dodaje do wynikowego kodu informacje niezbędne do pro
-lm	dołącza bibliotekę matematyczną (math.h)

Przykład:

```
$ gcc -Wall -Wextra -ansi -pedantic -fanalyzer liczba-err.c
```

Debugger w Visual Studio



GDB - debugger konsolowy projektu GNU

```
$ gcc -g -o program program.c
```

```
$ gdb program
```

```
(gdb) run
```

Źródło:  Visual Studio Debugging Seneca, Software testing

- **Analiza statyczna kodu** - analiza struktury kodu źródłowego lub kodu skompilowanego bez jego uruchomienia
- **lint, linter** - program do statycznej analizy
- wykorzystywane w automatyzacji weryfikacji jakości kodu
- zawierają reguły i heurystyki wykrywające podatności
- często koncentrują się na wybranym zestawie podatności dlatego warto stosować kilka rozwiązań
- uwaga: mogą generować fałszywie pozytywne ostrzerzenia
- analiza poprawności składni, detekcja luk bezpieczeństwa, weryfikacja stylu kodu, sugestie dotyczące wydajność, ...

Przykładowe narzędzia analizy statycznej:

- `cppcheck` - głównie wykrywanie krytycznych błędów
zob.: 📖 lista testów
`$ cppcheck --enable=all liczba-err.c`
- `clang-tidy` - wiele testów i bogate możliwości konfiguracji
zob.: 📖 lista testów
`$ clang-tidy -checks='*' liczba-err.c`

Narzędzia pozwalające dbać o styl

-  Artistic Style
\$ `astyle --style=kr nwd-balagan.c`
\$ `astyle --style=allman nwd-balagan.c`
-  ClangFormat
\$ `clang-format -style GNU nwd-balagan.c`
- on-line  formatter.org  Code Beautify

Przykłady wytycznych dotyczących stylu:

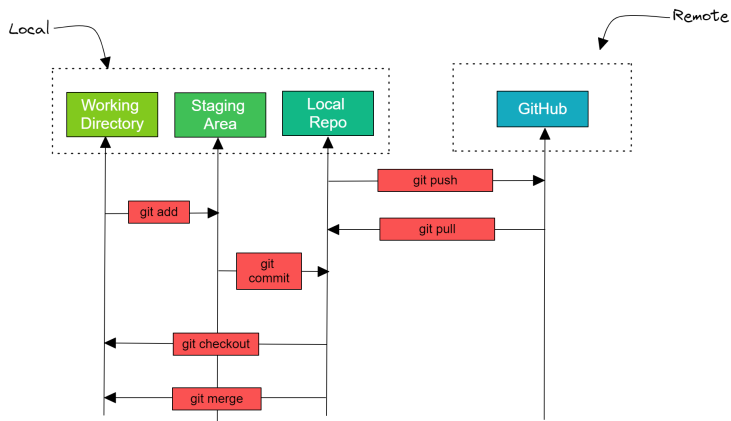
- wytyczne GNU  Making The Best Use of C
-  C Code Style Guidelines
-  Google C++ Style Guide

Git - rozproszony system kontroli wersji na licencji GPL, służący do zarządzania historią zmian w plikach i kodzie źródłowym projektów

- przechowuje historię zmian w kodzie
- umożliwia współdzielenie kodu w zespołach
- łatwe tworzenie i łączenie gałęzi
- synchronizacja historii pomiędzy repozytoriami, scalanie zmian
- szybki w działaniu - wiele operacji wykonywanych lokalnie
- elastyczny, wiele klientów i narzędzia

 GitHub - usługa chmurowa udostępniająca repozytoria git
Inne: GitLab, BitBucket, ...

<code>add</code>	dodanie pliku do rewizji (śledzenie zmian)
<code>commit</code>	zatwierdzenie zmian, powstaje nowy węzeł w historii repozytorium (lokalnie)
<code>push</code>	wypchnięcie zmian do zdalnego repozytorium
<code>pull</code>	pobranie zmian ze zdalnego repozytorium i scalenie z lokalną kopią
<code>merge</code>	scalanie zmian, łączenie gałęzi
<code>revert</code>	cofanie zmian
<code>reset</code>	przywracanie poprzednich stanów repozytorium
<code>log</code>	historia zmian
<code>diff</code>	porównywanie zmian w kodzie
<code>status</code>	aktualny stan repozytorium



Repozytorium z kodami z wykładu:

👉 https://github.com/IS-UMK/pp_wyklad

Klonowanie repozytorium

```
$ git clone https://github.com/IS-UMK/pp_wyklad
```

Historia zmian

```
$ git log
```

```
commit 01cc8a5dfb066e280ace0059ad05af9bbc951fbf (HEAD -> master, tag
```

```
Author: Marek <grochu@is.umk.pl>
```

```
Date:    Tue Oct 1 11:08:55 2024 +0200
```

Dodanie euclid1.f euclid1.pas

```
commit 0552f871c029186cca1784270411ce77aad5e266
```

```
Author: Marek <grochu@is.umk.pl>
```

```
Date:    Tue Oct 1 10:51:51 2024 +0200
```

Aktualizacja README

Dodanie pliku

```
$ git add nowy.c
```

Zapamiętanie zmian

```
$ git commit -m "Tu komunikat tłumaczący zmiany"
```

Wypchnięcie zmian do zdalnego repozytorium (wymagane uprawnienia do zmiany zawartości zdalnego repozytorium)

```
$ git push
```

Ściągnięcie i scalenie zmian pochopdzących ze zdalnego repozytorium

```
$ git pull
```