Narzędzia programistyczne

Przydatne narzędzia

- **kompolator** + 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 calu 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

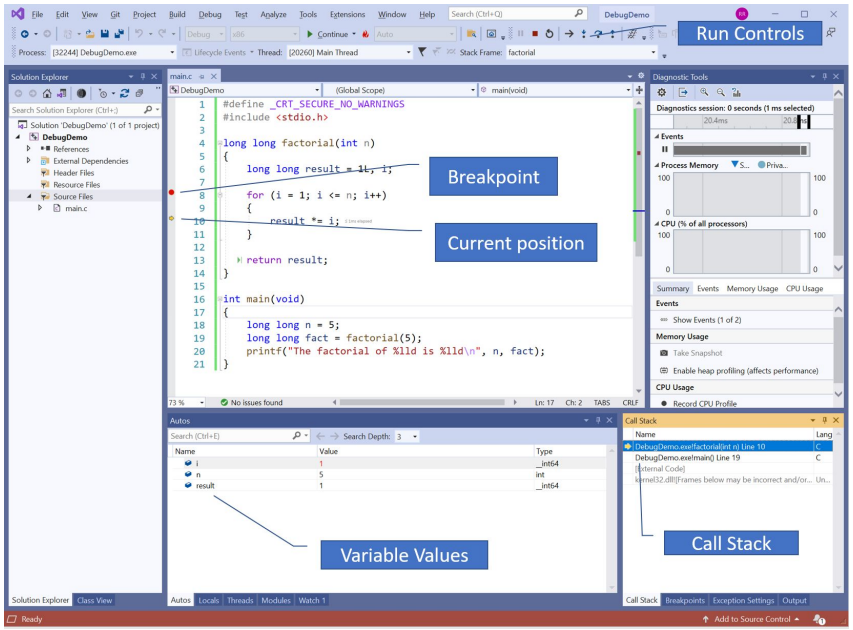
Przydatne opcje kompilatora GCC:

`-Wall`	włacza szereg reguł wykrywających potencjalne błedy, ge
`-Wextra`	włacza 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 - debuger 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
- czesto 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 wydajnośc, ...

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:

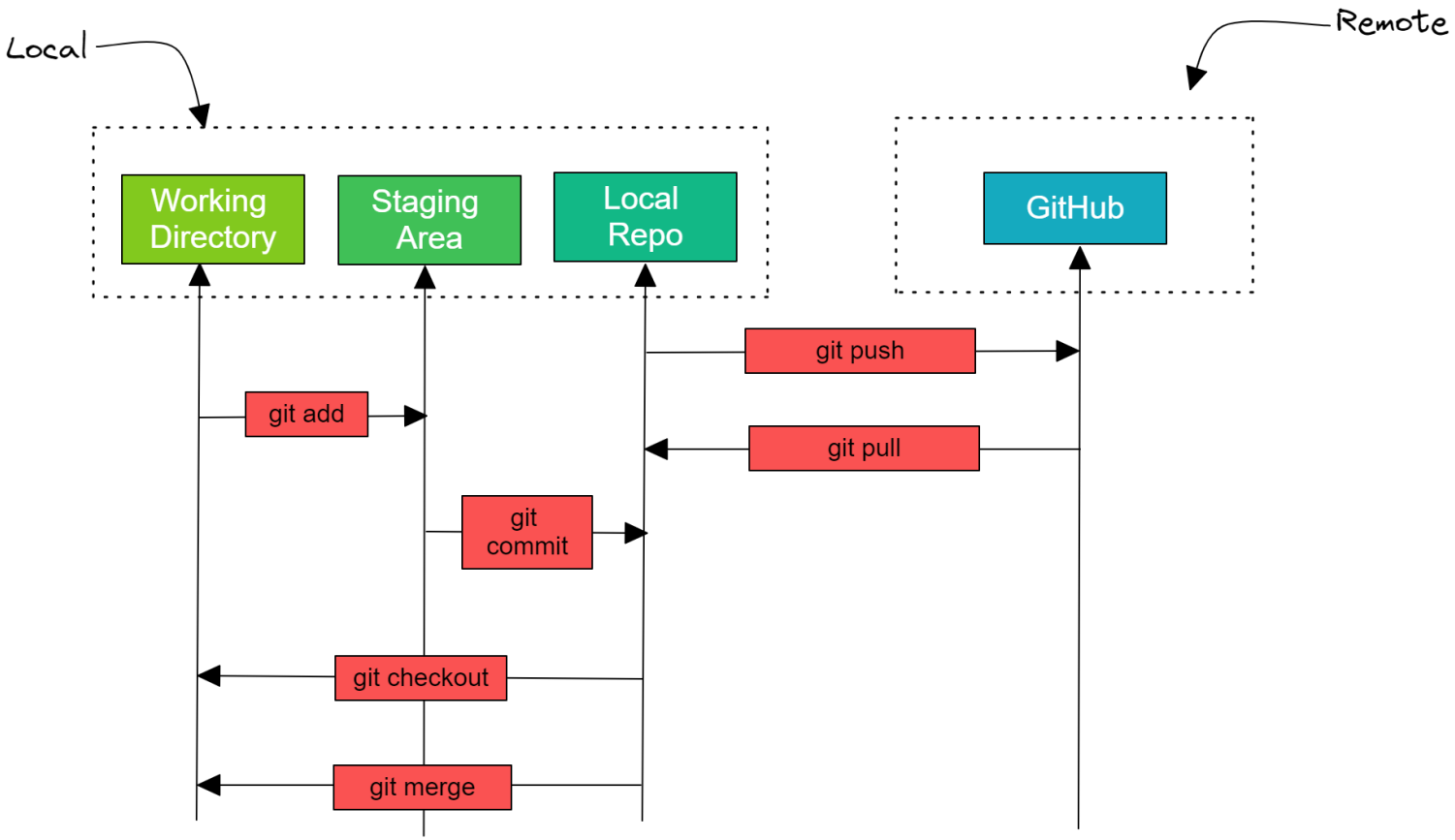
- wytyczne GNU ☞ Making The Best Use of C
- ☞ C Code Style Guidelines
- ☞ Google C++ Style Guide

- git - rozproszony system kontroli wersji na licencji GNU GPL
- zapamiętywanie śledzenie zmian w kodzie
- łatwe tworzenie i łaczenie gałęzi
- synchronizacja historii pomiędzy repozytoriami
- szybki w działaniu - wiele operacji wykonywanych lokalnie
- elastyczny, wiele klientów i narzędzia
- GitHub, GitLab, BitBucket - repozytoria git dostępne w chmurze

Najważniejsze komendy

- `add`: dodanie pliku do rewizji (śledzenie zmian)
- `commit`: zatwierdzenie zmian, powstaje nowy węzeł w historii repozytorium (lokalnie)
- `push`: wypchnięcie zmian do zdalnego repozytorium
- `pull`: pobranie zmian i scalenie z lokalną kopią
- tworzenie i scalanie (`merge`) gałęzi
- cofanie zmian (`revert`), przywracanie poprzednich stanów (`reset`)
- przeglądanie historii (`log`) i porównywanie zmian w kodzie (`diff`)

Etapy git



☞ *GitHub and Git Tutorial for Beginners*

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
```

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
```