Rozwiązywanie konfliktów w git

Do konfliktu dochodzi w momencie gdy

- Chcemy zrobić merge dwóch gałęzi
- Chcemy zrobić push lub pull, a werska lokalna i zdalna są skonfliktowane

Konflikt to sytuakcja, w której istnieją dwie różne wersje tekstu. Konieczność ręcznego rozwiązania konfliktu ma miejsce wtedy, gdy mechanizmy git nie są w stanie automatycznie połączyć dwóch różnych wersji.

W poniższym przykładzie git połączy obie wersje - ponieważ jest w stanie dokonać analizy tekstu i potrafi wykryć, że w tekście drugim dodano dodatkową linię. Wynikiem automatycznego scalenia będzie drugi tekst

Tekst1:

```
main(){
  cout<<"Paczki sa dobre"<<endl;
}
```

Tekst 2:

```
main(){
  cout<<"Precelki sa smaczne"<<endl;
  cout<<"Paczki sa dobre"<<endl;
}
```

W poniżym przykładzie git nie będzie w stanie automatycznie rozwikłać konfliktu, ponieważ nie wie, któa wersja jest prawidłowa:

Tekst1:

```
main(){
  cout<<"Paczki sa dobre"<<endl;
}
```

Tekst 2:

```
main(){
  cout<<"Paczki sa niesmaczne"<<endl;
}
```

Rezultatem tego konfliktu będzie utworzenie pliku o następującej zawartości:

```
main(){
<<<<<<< HEAD
  cout<<"Paczki sa dobre"<<endl;
=======
  cout<<"Paczki sa niesmaczne"<<endl;
>>>>>>> nazwaGalezi
}
```

To co jest między znacznikami `<<<<<<< HEAD` a `=======` to stan z naszego brancha zaś poniżej, pomiędzy `=======` a `>>>>>>> nazwaGalezi` to stan z gałęzi, z którą łączymy. W tej sytuacji musimy plik wyedytować, wybrać właściwą wersję i usunąć znaczniki.

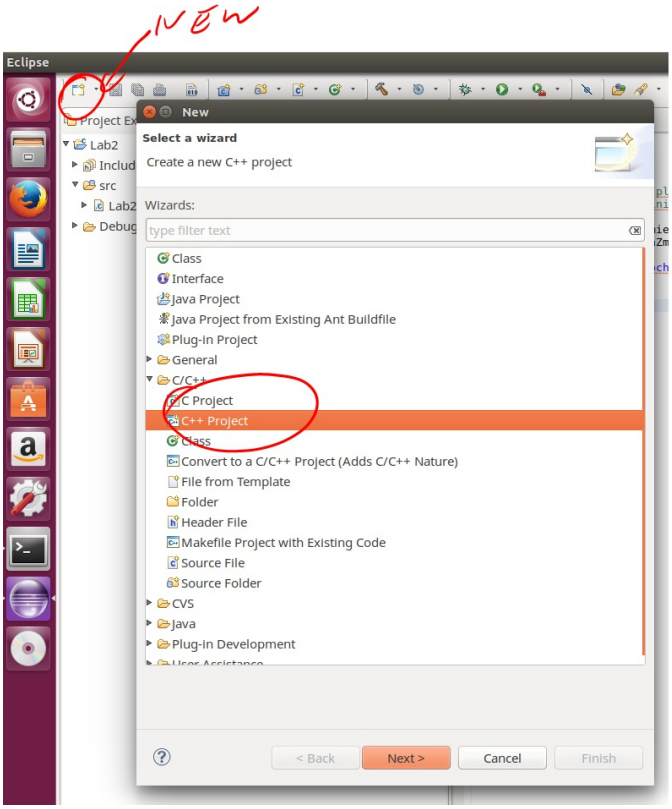
> Istnieją narzędzia wspomagające dobrym interfejsem

> usuwanie konfliktów jak na przykład meld lub p4merge. Wywołanie narzędzia poleceniem git mergetool - oczywiście po poprzednim zainstalowaniu narzędzia i skonfigurowaniem gita poleceniem git config --global merge.tool

Środowisko programistyczne

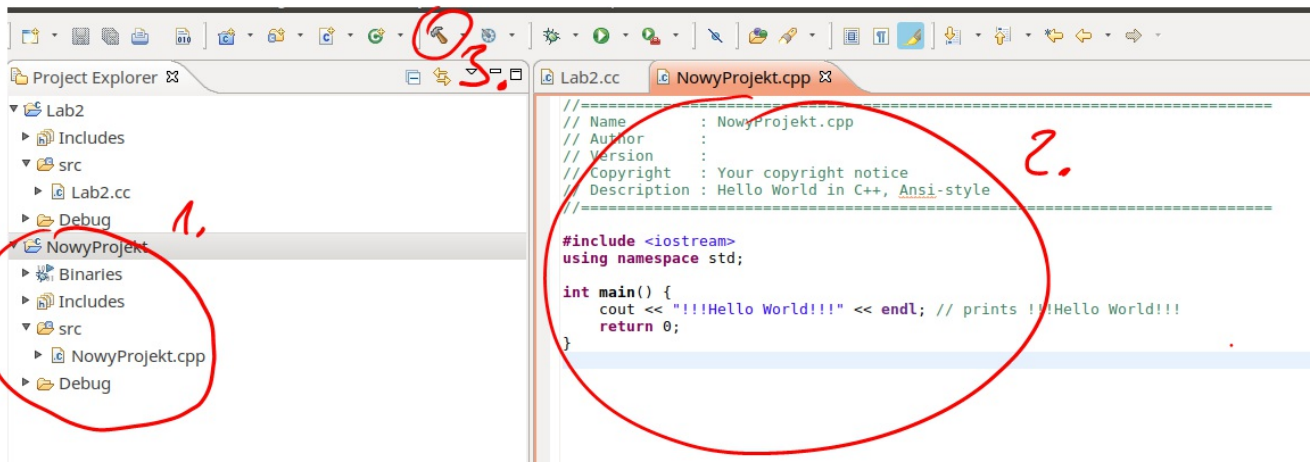
Środowisko programistyczne (IDE - Integrated Development Environment) ułatwia pisanie, formatowanie i debugowanie kodu. Środowisk jest bardzo wiele, w przyszłej pracy konkretne środowisko zapewni pracodawca. W tej instrukcji pokażemy, jak posługiwać się środowiskiem Eclipse. Eclipse to multiplatformowe (Windows, Linux...), modułowe środowisko umożliwiające programowanie w wielu językach. Jest dystrybuowane na darmowej licencji.

1. Kod w Eclipse zorganizowany jest w projekty. W ramach jednego projektu może funkcjonować wiele plików z kodem i innymi dokumentami (np daymi).
2. Eclipse domyślnie jest skonfigurowany do programowania w Javie. Chcąc programować np w C++ należy pobrać odpowiedni moduł (w przypadku C++ - C/C++ Devlopment Tools). Eclipse w laboratorium jest już odpowiednio skonfigurowany.
3. Eclipse domyślnie trzyma wszystkie projekty w katalogu ~/workspace
4. Aby stworzyć nowy projekt C++ należy wybrać New, z okna dialogowego C/C++->C++ Project

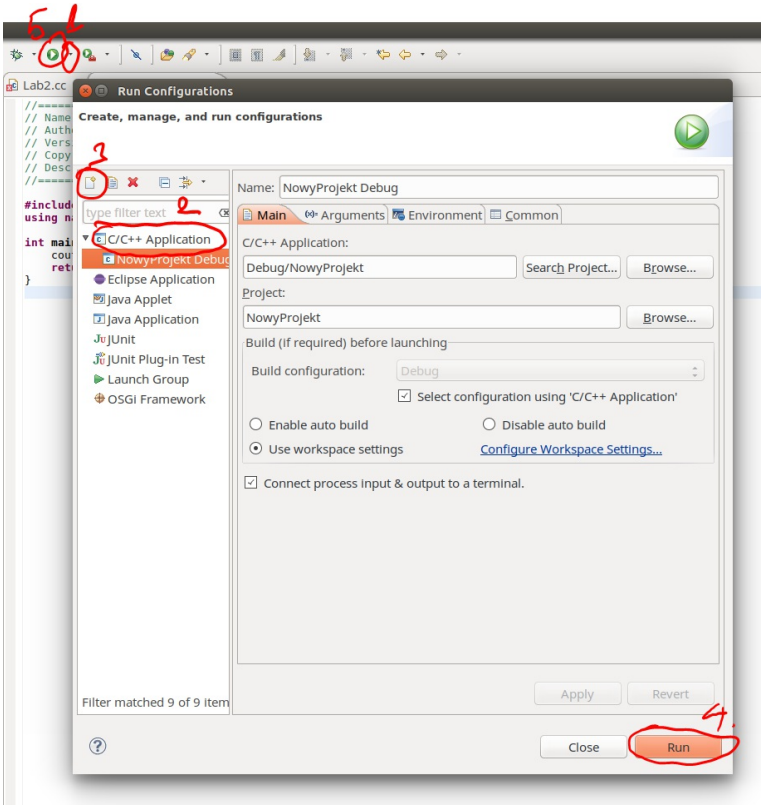


5. W drugim kroku należy podać nazwę projektu (1), wybrać typ projektu (program wykonywalny i typ - na poczatek "Hello World C++ Project") (2) . Wskazany powinien być zestaw narzedzi (Toolchain) "LinuxGCC" (3).

6. Po lewej stronie (1) zobaczymy katalogi projektu - z istotnych "src" - zawiera kod projektu zaś "Debug" - skompilowany progam i pliki towarzyszące. Po prawej - kod (2). Przycisk z "młotkiem" uruchomi kompilację.



7. Aby uruchomić skompilowany problem należy przygotować konfigurację uruchomienia. "Trójkącik w dół" (1), "Run Configurations", następnie należy wybrać C/C++ Application (2), przycisk "New" (3) i "Run" (4). Program się uruchomi w konfiguracji "Debug", wynik pojawi się na konsoli Eclipse. Od tej pory program można uruchamiać domyślnie w tej konfiguracji przyciskiem "Play" (5).



> Programując stosuje się dwie zasadnicze konfiguracje. W konfiguracji "Debug" do kodu na etapie kompilacji dołączane są znaczniki debugera, umożliwiające jego działanie. Kod wynikowy kompilatora nie jest optymalizowany dla szybkości działania. Kod powstały w tej konfiguracji może być używany w debugerze (o tym później) ale będzie wolny, choć będzie kompilował się szybko. W konfiguracji "Release" program nie będzie współpracował z debugerem ale kod będzie optymalizowany - będzie kompilował się wolno, wykonywał - szybko.

Tablice

1. Proszę skompilować kod Lab5_1.cpp. Czy kod wykonuje się poprawnie? Proszę zmienić kod tak, aby wyświetlać jedynie zawartość tabel. Dla lepszego zrozumienia problemu proszę dopisać nieskończoną pętlę while wypisującą kolejne elementy tablicy w postaci id: , wartosc:. Proszę zobaczyć, że program zawsze zgłosi błąd. Rozwiązanie proszę zapisać jako plik Lab5_1_solved.cc i umieść w swojej gałęzi git.
2. Proszę napisać program który wypełnia tablicę kwadratami liczb od 1 do 10. Następnie proszę wypisać kwadraty liczb występujących w tablicy tab[] = {1, 5, 3, 10};. Proszę zachować kolejność z tablicy tab. Rozwiązanie proszę zapisać jako plik Lab5_2_solved.cpp i umieść w swojej gałęzi git.
3. Proszę napisać program w którym deklarowana jest tablica intów. Następnie proszę obliczyć i wypisać, sumę, minimum, maksimum i wartość średnią wszystkich elementów tablicy. Rozwiązanie proszę zapisać jako plik Lab5_3_solved.cpp i umieść w swojej gałęzi git.
4. Proszę stworzyć tablicę dwuwymiarową z wynikami tabliczki mnożenia liczb od 1 do 10. Proszę wypisać ją na ekran, może być pomocny znak tabulacji \t. Rozwiązanie proszę zapisać jako plik Lab5_4_solved.cpp i umieść w swojej gałęzi git.
5. Proszę rozbudować kod Lab5_5.cpp, dopisując przypisanie zmiennej au do drugiego elementu tablicy oraz obliczając długość tablicy z wykorzystaniem funkcji sizeof(). Rozwiązanie proszę zapisać jako plik Lab5_5_solved.cpp i umieść w swojej gałęzi git.
6. Proszę policzyć liczbę wystąpień każdej cyfry w tablicy digits w pliku Lab5_6.cpp. Rozwiązanie proszę zapisać jako plik Lab5_6_solved.cpp i umieść w swojej gałęzi git.

Wskaźniki

1. Proszę zmodyfikować kod Lab5_7.cpp, dodając opis, co znaczą wypisane liczby. Dodatkowo proszę dopisać wypisanie wartości elementów tablicy. Proszę zmienić tablicę z int na double oraz char (dla zmiennej char trzeba przy wyświetleniu rzutować wskaźnik za pomocą (void *)) i zobaczyć co się zmieni. Rozwiązanie proszę zapisać jako plik Lab5_7_solved.cpp i umieść w swojej gałęzi git.
2. Które z przypisań w pliku Lab5_8.cpp są poprawne. Proszę wykomentować błędne linie kodu oraz dopisać komentarz do każdego przypisania. Proszę wypisać:

- wartości zmiennej,
- adres zmiennej,
- adres wskazywany przez wskaźnik
- wartość wskaźnika.

Rozwiązanie proszę zapisać jako plik Lab5_8_solved.cpp i umieść w swojej gałęzi git.

3. Proszę napisać kod, w którym zmienna jest zmodyfikowana z wykorzystaniem wskaźnika, a nie samej zmiennej. Proszę wypisać wartość zmiennej przed i po modyfikacji. Wypisanie też powinno wykorzystać wskaźnik. Rozwiązanie proszę zapisać jako plik Lab5_9_solved.cpp i umieść w swojej gałęzi git.
4. Proszę napisać program, w którym mamy zadeklarowaną tablicę char, która zawiera napis posiadający minimum dwa słowa. Następnie z wykorzystaniem wskaźników do char proszę wypisać cały napis oraz z pominięciem pierwszego słowa. Rozwiązanie proszę zapisać jako plik Lab5_10_solved.cpp i umieść w swojej gałęzi git.
5. Proszę zmodyfikować kod Lab5_5_solved.cpp i wykorzystać wskaźniki wraz z odpowiednim sposobem odwołania się do

struktur.

6. Proszę sprawdzić co się stanie, jeżeli wskaźnik wskazuje zmienną, której nie ma. Proszę wykorzystać deklarację zmiennej wewnątrz pętli lub warunku. Rozwiązanie proszę zapisać jako plik Lab5_11_solved.cpp i umieść w swojej gałęzi git.
7. Proszę napiać kod wypisujący elementy tablicy z wykorzystaniem inkrementacji wskaźników. Proszę przy okazji wypisywania elementów tablicy wypisać adresy wskazywane przez wskaźnik. Proszę zrobić to dla dwóch różnych typów i własnej strukury. Rozwiązanie proszę zapisać jako plik Lab5_12_solved.cpp i umieść w swojej gałęzi git.
8. Proszę napisać program zmieniający kolejność elementów w tablicy (pierwszy staje się ostatnim, drugi przedostatnim itd.) z wykorzystaniem wskaźników. Proszę zrobić to dla dwóch różnych typów i własnej struktury. Rozwiązanie proszę zapisać jako plik Lab5_13_solved.cpp i umieść w swojej gałęzi git.
9. **Dla ambitnych.** Proszę zadeklarować tablicę wskaźników, w której każdy wskaźnik wskazuje na coraz większą tablicę (od 1 do 4 elementów). Tablica z 4 elementami powinna mieć pewne liczby, tablica z trzema elementami posiada dwa pierwsze elementy tablicy większej i sumę dwóch ostatnich elementów. Podobnie tablica o długości 2 i 1. Proszę zainicjalizować tą tablicę oraz wyświetlić ją z wykorzystaniem wskaźników. Rozwiązanie proszę zapisać jako plik Lab5_14_solved.cpp i umieść w swojej gałęzi git.
10. W której linijce programu Lab5_15.cpp jest błąd i na czym on polega. Proszę poprawnie wyświetlić wszystkie zmienne. Rozwiązanie proszę zapisać jako plik Lab5_15_solved.cpp i umieść w swojej gałęzi git.