WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Zadanie 2a. Pomiar czasu.

Problem 1: Chcemy zmierzyć aktualny czas z dokładnością do milisekund lub lepszą.

Problem 2: Chcemy zmierzyć, ile czasu zajmuje wykonanie funkcji (lub ogólnie XX kolejnych instrukcji).

Zadanie: Proszę zaproponować funkcję/klasę do pomiaru czasu z dokładnością do milisekund lub lepszą (mikrosekundy).

Proponowana metoda pomiaru: zapamiętanie znacznika czasowego (czasu systemowego) przed wywołaniem funkcji oraz po wywołaniu funkcji, obliczenie różnicy (mile widziane inne własne pomysły).

```
double start = _m_timestamp();
// wykonuj cos czasochlonnego
double period  = _m_timestamp() - start;   // przykladowa wartość period = 2.12 czyli 2 sekundy i 120 ms
```

Wynik: jako wartość double: liczba sekund (od początku roku 1970) przed przecinkiem, po przecinku zaś liczba milisekund (mikrosekund itd.), lub jako typ całkowity zawierający liczbę milisekund, lub jako prosta struktura/obiekt (wtedy potrzebna jest funkcja odejmująca dwa znaczniki czasowe).

Założenie: Wykorzystane funkcje/biblioteki nie powinny ograniczać wyboru platformy (kod powinien się kompilować i uruchamiać zarówno w systemie Windows jak i Linux)

Dodatkowe zadanie (opcja):
Pobierz czas systemowy w postaci ROK, MIESIĄC, DZIEŃ, GODZINA, MINUTA, SEKUNDA, MILISEKUNDA
Przedstaw czas w postaci std::string, np. „2023.03.01.00.05.45.004” (1 marca tego roku, 5 minut po północy, 45 s, 4 ms).
Pomijamy czas zimowy/letni i konsekwencje wynikające ze zmiany czasu (ale mamy świadomość ich istnienia).

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Zadanie 2b. Funkcje do konwersji typów.

1. Zaproponuj zestaw funkcji do konwersji **typ_calkowity -> std::string** oraz **std::string -> typ_calkowity**

2. J/w, ale zestaw funkcji do konwersji **typ_zmiennoprzecinkowy -> std::string** oraz **std::string -> typ_zmiennoprzecinkowy**

Np.:
_c2s (char parametr) – char to std::string
_i2s (int parametr) – int to std::string
_ui2s (unsigned int parametr) – unsigned int to std::string
_d2s (double parametr, int liczba_cyfr_po_przecinku) – double to std::string – z zadaną precyzją (jako drugi parametr)
_s2f (std::string parametr) – std::string to float
_s2l (std::string parametr) – std::string to long

Dodatkowa opcja:
- Konwersja typów całkowitych (raczej unsigned) na std:: string w postaci szesnastkowej (12345678 -> „bc614e”)
- J/w z zadaną minimalną liczbą znaków wyniku, np. _l2sx (12345678, 8) oznacza „long to std:: string hex”, razem ma być nie mniej niż 8 znakow (uzupełnienie z przodu neutralnym znakiem ‚0’) -> „00bc614e”
- Konwersja std:: string zawierającego postać szesnastkową na typ całkowity _sx2l (std::string hex to unsigned long)

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Zadanie 2c. Biblioteka statyczna.

1. Proszę zapoznać się z pojęciami (w języku C/C++): biblioteka statyczna i biblioteka dynamiczna, jak się je wykorzystuje i jakie są różnice. Jak utworzyć projekt klasy statycznej w Code::Blocks ?

2. Proszę zaproponować (napisać i skompilować) bibliotekę statyczną, w której zawarte będą funkcje z zadania 2a i 2b.

3. Proszę zmodyfikować projekt z zadania 1 (wykorzystujący statyczną klasę diagnostyczną) w taki sposób, żeby korzystał z tej biblioteki statycznej (wystarczy, że biblioteka będzie użyta z sukcesem w głównym pliku main.cpp).

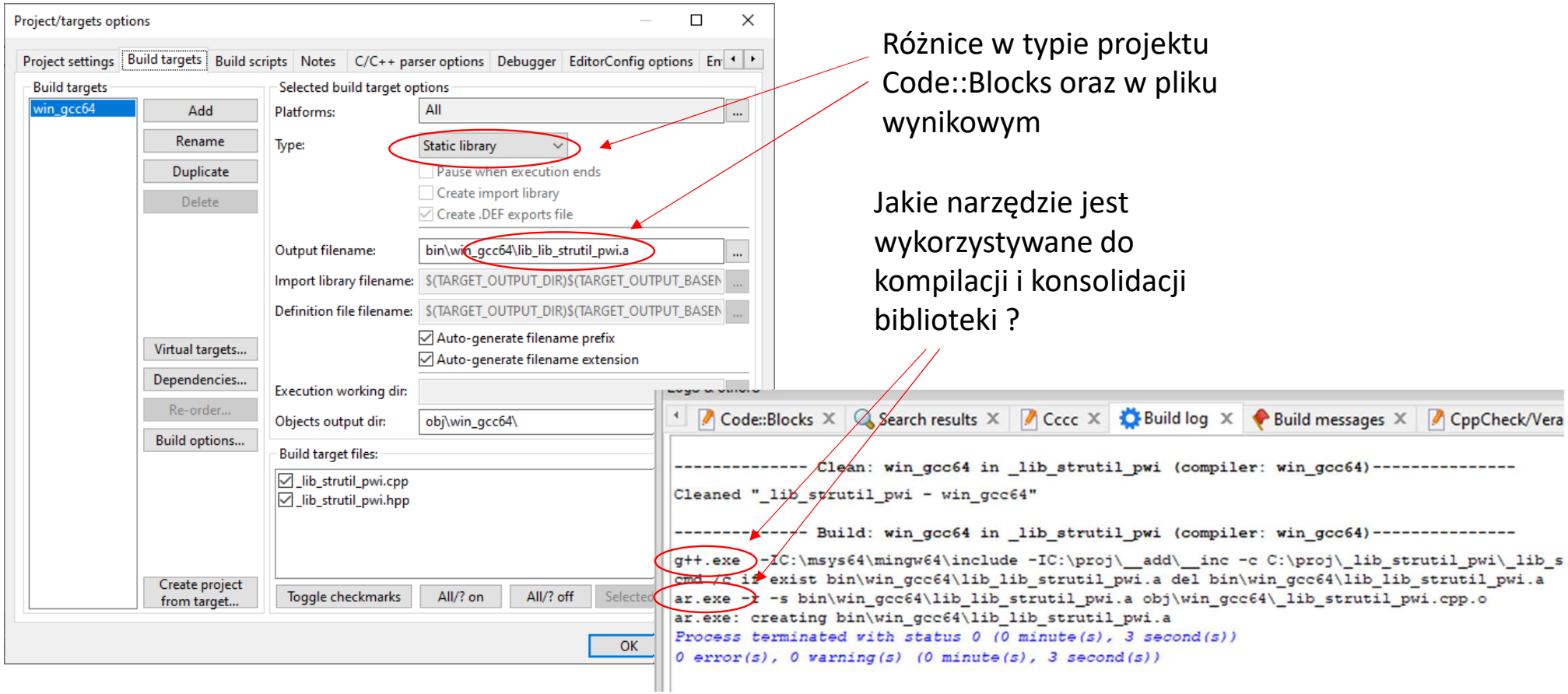
4. Czy wspomnianą statyczną klasę diagnostyczną z zadania 1 możemy zamknąć w bibliotece statycznej zachowując funkcjonalność związaną z **DIAG_ENABLE** ? (zmiana treści makra uzależniona od tego, czy stała DIAG_ENABLE jest zdefiniowana)

Zadanie dodatkowe jest opisane na końcu materiałów pomocniczych.

Do e-maila dołączam plik 7zip ze spakowanym projektem/szablonem w Code::Blocks.

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Materiały do zadania 2c Biblioteka statyczna



Różnice w typie projektu Code::Blocks oraz w pliku wynikowym

Jakie narzędzie jest wykorzystywane do kompilacji i konsolidacji biblioteki ?

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Materiały do zadania 2c Biblioteka statyczna

Ręczna kompilacja i zbudowanie najprostszej biblioteki statycznej (z istniejących plików .hpp i .cpp):

1. Uruchom terminal i przejdź do folderu z plikami biblioteki (np. „C:\proj_lib_strutil_pwi")
2. Wykonaj po kolei komendy: (co oznaczają?)

```
@path = %path%;C:\msys64\mingw64\bin;
del _lib_strutil_pwi.o
del lib_lib_strutil_pwi.a
g++ -c _lib_strutil_pwi.cpp
ar rcs lib_lib_strutil_pwi.a _lib_strutil_pwi.o
```

3. Na potrzeby przyszłych projektów wykorzystujących tą bibliotekę:
- skopiuj ręcznie plik biblioteki lib*.a do folderu dostępnego dla linkera (np. „C:\proj__add__lib_win_gcc64")
- skopiuj ręcznie plik nagłówka *.hpp do folderu dostępnego dla kompilatora (np. „C:\proj__add__inc")

Zadanie 2c Opcja (dodatkowe):

Zaproponuj plik makefile z 3 regułami:
- clean : czyści pliki pośrednie i wynikowe
- build : budowanie biblioteki
- all (domyślna) : czyści a potem buduje (wykorzystuje dwie poprzednie reguły) na koniec kopiuje pliki jak w punkcie 3 (powyżej)

Spróbuj zbudować bibliotekę za pomocą narzędzia make (mingw32-make).

Czy w opcjach projektu w Code::Blocks można skonfigurować automatyczne kopiowanie plików z punktu 3 po zbudowaniu projektu ?