C - 0 - Wprowadzenie

Środowiskiem dla wszystkich ćwiczeń będzie Visual C++ - wersja 6.0. Mimo że Visual w nazwie zawiera „C++", początkowo będziemy pisać programy w „gołym" języku C. Następnie przejdziemy do języka C++, zaś na końcu wykonamy kilkanaście programów okienkowo-rysunkowych.

Proszę wykonywać wszystkie zadania i quizy.

0.1 Założenie projektu konsolowego

0. Otwórz Visuala klikając na ikonkę  .

1. Wybierz z menu opcję **File->New**.

2. Na ekranie pokaże się mniejsze okienko o tytule **New** - upewnij się, że wybrano drugą zakładkę, czyli zakładkę **Projects**.

3. Z listy po lewej stronie wybierz opcję **Win32 Console Application** - z reguły jest to trzecia opcja od dołu.

4. Nadal w tym samym okienku, z prawej strony u góry wpisz nazwę swojego projektu.

Nigdy, przenigdy, nie używaj polskich liter ani spacji !!!

5. Naciśnij **OK** - pojawi się kolejne okno, w którym należy wybrać opcję **A „Hello, World!" application** i nacisnąć **Finish**. Pojawi się jeszcze jedno małe okienko – naciśnij **OK**.

6. Projekt wraz z plikiem o nazwie postaci **[nazwaprojektu].cpp** jest już utworzony - znajduje się w drzewku po lewej stronie. Rozwiń to drzewko, następnie rozwiń folder **Globals** i kliknij dwa razy na różowy romb, by w głównym oknie pojawiła się treść programu.

7. Zapisz od razu swój projekt - służy do tego ikonka z kilkoma dyskietkami (czwarta od brzegu).

Nigdy, przenigdy, nie korzystamy z SaveAs !!!
Zawsze korzystamy z Save -po skorzystaniu z SaveAs różne części projektu zapisują się w różnych miejscach i nie mogą się potem odnaleźć.

0.2 Kompilacja i uruchomienie

Należy kliknąć ikonkę z wykrzyknikiem lub nacisnąć **Ctrl+F5** – pojawi się czarne okienko z napisem **Hello world** i podpowiedzią (po angielsku), że należy nacisnąć dowolny klawisz by kontynuować – naciśnij cokolwiek.

0.3 Zawartość pliku

Twój pierwszy program wygląda mniej więcej następująco:

```
// mojpierwszyprojekt.cpp : Defines the entry point for th
//

#include "stdafx.h"

int main(int argc, char* argv[])
{
    printf("Hello World!\n");
    return 0;
}
```

Komentarz z nazwą projektu.

Deklaracja pliku nagłówkowego – tzw. biblioteki.

Funkcja main – najważniejsza funkcja w programie – każdy program musi ją zawierać.

Komentarze można pisać na dwa sposoby:

// to jest komentarz jednolinijkowy

/* jeżeli mamy do zakomentowania wiele linijek, to otaczamy je ukośnikami i gwiazdkami, ponieważ zakomentowanie każdej z wielu linijek za pomocą dwóch ukośników byłoby bardzo męczące dla programisty, nie wspominając już o tym, że byłoby to po prostu nieprofesjonalne i niechlujne ... */

90% studentów stosuje tylko pierwszy sposób komentowania, nawet jeżeli do zakomentowania mają 20 linijek lub więcej :(

Polecenie **#include** dołącza do programu pliki nagłówkowe (czyli biblioteki), które zawierają dodatkowe funkcje umożliwiające np:

- obsługę działania na plikach (zapis/odczyt)
- obsługę grafiki
- manipulowanie łańcuchami tekstowymi

Biblioteka **stdafx.h** jest biblioteką endemiczną dla Visuala. W innych środowiskach zamiast tej biblioteki dołączylibyśmy zapewne **stdio.h** lub **iostream.h**. Inne najczęściej dołączane biblioteki, to:
- **conio.h** – dodatkowa obsługa klawiatury i monitora
- **string.h** – obsługa łańcuchów tekstowych
- **math.h** – funkcje matematyczne

Funkcję **main** musi zawierać każdy program. Słowo **int** (skrót od **integer**) oznacza, że funkcja **main** zwraca wartość całkowitoliczbową. Przyjęło się, że w razie sukcesu funkcja ta zwraca wartość **0**. Jeżeli jednak wystąpił jakiś błąd, to zwraca się inną wartość (programista decyduje jaką) – najlepiej, by była to inna wartość dla każdego rodzaju błędu, ponieważ pomaga to w diagnostyce błędów programu.

Obok funkcji **main** występują w nawiasach dwie wartości:
- **int argc** – liczba argumentów podanych z linii poleceń
- **char* argv[]** – tablica tych argumentów

Będziemy korzystać z tych zmiennych w późniejszych ćwiczeniach przy okazji tablic - na razie nie należy przejmować się nimi.

Funkcja **main** ma dwie linijki zawartości:
- polecenie **printf** drukuje na ekranie wiadomość ujętą w cudzysłowach. Znaczek **\n** oznacza, że po wydrukowaniu napisu chcemy przejść do nowej linii.
- polecenie **return 0;** zwraca wartość **0**, co oznacza, że uznajemy, iż działanie funkcji **main** zakończyło się sukcesem (trudno uznać wydrukowanie czegoś na ekranie za błąd).

Zauważ, że instrukcje wykonywane przez program kończą się **średnikiem**.

0.5 Zadania

Zadanie 1. Zaraz pod linijką drukującą napis wpisz taką samą linijkę. Skompiluj i uruchom program. (Jeżeli pojawi się małe okienko, naciśnij **Yes**). **Po zakończeniu zadania zawsze zamykaj czarne okienko**.

Zadanie 2. Teraz w pierwszej linijce drukującej napis usuń znaczek **\n**. Skompiluj i uruchom program, zauważ różnicę. Wpisz pomiędzy **Hello** i **World** znaczek **\t** i zauważ, że wypisuje on tabulację.

Zadanie 3. Rozbij pierwszą linijkę drukującą napis na dwa **printf**y i sprawdź, czy napis nadal wygląda tak samo – nie zgub spacji między **Hello** i **World**.

Zadanie 4. Sprawdź, że **\n** nie musi być na końcu linijki – wpisz go w środek drukowanego napisu i zobacz efekty. Możesz też go wpisać na początku, zaraz po cudzysłowie.

Zadanie 5. Po pierwszej linijce drukującej napis wpisz **return 0;**. Uruchom program. Zauważ, że program zakończył swoje działanie po pierwszym **printf**ie **–** gdy każemy funkcji zwrócić wartość, to kończy ona swoje działanie, nawet gdy dalej są jeszcze jakieś instrukcje.

0.6 Próby zepsucia programu

Zadanie 6. Zakomentuj (za pomocą dwóch ukośników) na chwilę linijkę zwracającą wartość funkcji **main**. Skompiluj i uruchom program. Program uruchomi się, ale na dole Visualowego okna pojawi się 1 **warning** (ostrzeżenie), przewiń trochę do góry dolną część okienka, by przeczytać treść tego warninga – powinna brzmieć mniej więcej tak:
warning C4508: 'main' : function should return a value; 'void' return type assumed
Oznacza to tyle, że funkcja powinna zwracać wartość, ale dla świętego spokoju Visual przyjął, że jest to funkcja typu **void** (ang. próżnia, pustka), czyli funkcja niezwracająca żadnej wartości (tylko np. coś robiąca).
Odkomentuj z powrotem zakomentowaną linijkę zwracającą wartość funkcji main.

Zadanie 7. Zakomentuj (za pomocą dwóch ukośników) na chwilę linijkę dołączającą bibliotekę stdafx.h. Skompiluj i uruchom program. Tym razem na dole Visualowego okna pojawi się 1 **error** – czyli błąd. O ile z warningami program może działać, to z errorami już się nie uruchomi. Przeczytaj treść errora, będzie brzmieć następująco:
fatal error C1010: unexpected end of file while looking for precompiled header directive
Po polsku: podczas poszukiwania pliku nagłówkowego (czyli tego zakomentowanego **stdafx.h**) napotkano niespodziewanie na koniec pliku. Odkomentuj z powrotem tę linijkę.

To, że program uruchamia się mimo warningow nie oznacza, że można te warningi tak po prostu zostawiać. Należy dążyć do tego, by program nie zgłaszał ani errorów ani warningów.

Zadanie 8. Wykasuj średnik na końcu pierwszej funkcji drukującej napis na ekranie. Skompiluj i uruchom program. Tym razem pojawi się błąd:
error C2146: syntax error : missing ';' before identifier 'printf'
Przed **printf** (tym drugim **printf**) brakuje średnika – Visual zwraca uwagę **przed czym** brakuje średnika, a nie **po czym** brakuje średnika.
Kliknij dwa razy na opis błędu na dole Visualowego okna, Visual zaznaczy wówczas linijkę, w której jest błąd. Bardzo często błędu należy szukać właśnie linijkę wyżej.

Studenci nagminnie szukają tej linijki wzrokiem, zamiast od razu klikać na informację o błędzie :(

Po wykonaniu eksperymentu przywróć wykasowany średnik.

Po poznaniu nowego elementu języka programowania warto trochę popsuć program i przeczytać, jakie błędy się wyświetlą i dlaczego właśnie takie.

0.7 Gdzie jest projekt?

Projekt domyślnie zapisuje się w:
C:\Program Files\Microsoft Visual Studio\MyProjects\[katalog o nazwie projeku]

Doklikaj się do projektu i obejrzyj zawartość katalogu.
Plik z treścią programu ma rozszerzenie **cpp** i znajduje się od razu w katalogu z projektem, zaś skompilowany program z rozszerzeniem **exe** znajduje się w podkatalogu **Debug**. Na razie nie ma sensu go uruchamiać przez klikanie – możesz sprawdzić, że pod większością „nowych" Windowsów program najprawdopodobniej tylko „mignie" i od razu zamknie się.

Jak to naprawić?
1. Zaincluduj w swoim programie bibliotekę **conio.h** (zawiera ona funkcję **getch()**).
2. Przed **return**em dopisz **getch();** - instrukcja ta zatrzymuje program i czeka na naciśnięcie klawisza.
3. Skompiluj program.
4. Doklikaj się do swojego projektu i uruchom jeszcze raz swój program z rozszerzeniem **exe** (jest w podkatalogu **Debug**).

Uwaga – Visual może czasem (na szczęście rzadko) nie reagować na zmiany wprowadzane w programie, należy wówczas ręcznie wykasować podkatalog **Debug**, tak by Visual musiał stworzyć go od nowa. Może też pomóc kompilacja przez wybranie z menu opcji **Build->Rebuild All**.

0.8 Jak otworzyć stary projekt?

Wybierz z menu **File->Open Workspace** i doklikaj się do projektu, kliknij na plik z rozszerzeniem **dsw** i kliknij **Open**.

Drugi sposób: otwórz Visuala, otwórz osobno katalog ze swoim projektem. Złap za plik z rozszerzeniem **dsw** i przeciągnij go w okno Visuala.

Studenci nagminnie klikają na wszystkie możliwe pliki, byle nie dsw i dziwią się, że projekt się nie ładuje. Bardzo często też wybierają File->Open zamiast File->Open Workspace :(

Zamknij Visuala, otwórz go ponownie i załaduj swój projekt.

0.9 Częste błędy

Zapominanie o średniku na końcu instrukcji.

Pisanie **/n** zamiast **\n** .

Gubienie klamerek **{** oraz **}** w ferworze pisania programu.

Gubienie nawiasów okrągłych **(** oraz **)** i cudzysłowów.

Zapominanie o zwróceniu wartości funkcji (**return** ...).

Nie zamykanie czarnego okienka po skończeniu podziwiania programu – wówczas program może się nie skompilować na nowo, ponieważ plik *.**exe** (czyli czarne okienko) jest uruchomiony i dostęp do niego

jest zablokowany.

Używanie polskich liter i spacji w nazwach plików i/lub projektów.

Używanie **SaveAs** do zapisania projektu zamiast ikonki z kilkoma dyskietkami.

Nie czytanie błędów i oczekiwanie od prowadzącego zajęcia, że to on je przeczyta i wyjaśni – podstawowych błędów NIE PRZECZYTAM I NIE WYJAŚNIĘ!!! W przypadkach rażącego nieprzygotowania do zajęć obniżę punkty za zadanie.

Otwieranie starego projektu w zły sposób.

0.10 Quiz

1. Jakie słowo kluczowe dołącza bibliotekę do programu?
   _ _ _ _ _ _ _
2. Jaką funkcję musi zawierać program?
   _ _ _ _
3. Jakie słowo kluczowe zwraca wartość funkcji?
   _ _ _ _ _ _
4. Jakimi znakami otaczamy napis do wydrukowania na ekranie?
   _
5. Za pomocą jakiego znaczka przechodzimy do następnej linii?
   _ _
6. Za pomocą jakiego znaczka piszemy tabulację?
   _ _
7. Jak się nazywa typ całkowitoliczbowy?
   _ _ _
8. Która z nazw projektu jest prawidłowa i dlaczego?
   a) mójProjekt b) moj projekt c) mojProjekt d) mój projekt
   _____
9. Jaki znak interpunkcyjny kończy instrukcję?
   _
10. Jaka funkcja wypisuje napis na ekranie?
    _ _ _ _ _ _
11. Co jest groźniejsze, warning, czy error?
    _____
12. Jak można odnaleźć linijkę z błędem?
    _____
13. Jak nie należy zachowywać programu?
    _____
14. Jakie rozszerzenie ma plik, dzięki któremu otwieramy stary projekt?
    _ _ _
15. W jakim podkatalogu znajduje się plik wykonywalny z rozszerzeniem *.**exe**?
    _ _ _ _ _
16. W jakiej bibliotece znajduje się funkcja **getch()** i co ona robi?
    _ _ _ _ _ . h
    _____
17. Masz do zakomentowania 23 linijki. Oblicz, ile razy musisz nacisnąć klawisze /, by zakomentować taką liczbę linijek dwoma ukośnikami.
    _____
    _____
    _____
    _____
    Ile razy musisz nacisnąć klawisze / oraz * by zakomentować 23 linijki drugim sposobem?
    _____
    _____
    _____
18. Co się stanie, jeżeli w zwrócenie wartości funkcji nie będzie ostatnią instrukcją tej funkcji?
    _____
    _____
    _____