Politechnika Częstochowska
Wydział Inżynierii Mechanicznej i Informatyki

Laboratorium z przedmiotu
Bezpieczeństwo komunikacji elektronicznej

Funkcje skrótu (hashihg functions)
Algorytmy MD5 i SHA256

1. Cel ćwiczenia laboratoryjnego

Ćwiczenie ma na celu objaśnienie zasad działania oraz zastosowania w metodach kryptograficznych tzw. funkcji skrótu, znanych pod nazwą funkcji „hashujących" (z ang. hashing functions).

2. Funkcje skrótu i ich zastosowania w kryptografii

Funkcja skrótu, **funkcja mieszająca** lub **funkcja haszująca** – to funkcja przyporządkowująca dowolnie dużej liczbie krótką, zawsze posiadającą stały rozmiar, niespecyficzną, quasi-losową wartość, tzw. *skrót nieodwracalny*. Stosowano również nazwę **funkcja jednokierunkowa**, co oznacza, że stosunkowo łatwo jest obliczyć jej wartość dla danej liczby, natomiast znalezienie przekształcenia odwrotnego (funkcji odwrotnej) pozwalającego na odtworzenie danej liczby mając jej skrót – jest praktycznie niemożliwe.

W informatyce funkcje skrótu pozwalają na ustalenie krótkich i łatwych do weryfikacji sygnatur (*fingerprint* – odcisk palca) dla dowolnie dużych zbiorów danych. Sygnatury mogą chronić przed

przypadkowymi lub celowo wprowadzonymi modyfikacjami danych (sumy kontrolne), a także mają zastosowania przy optymalizacji dostępu do struktur danych w programach komputerowych (tablice mieszające).

Szczególną podgrupą funkcji skrótu są funkcje uznawane za bezpieczne do zastosowań kryptologicznych (jak np. SHA-3). Kryptograficzna funkcja skrótu powinna spełniać kombinację następujących kryteriów, w zależności od zastosowania:

1. Odporność na kolizje (*collision resistance*) – brak praktycznej możliwości wygenerowanie dwóch dowolnych wiadomości o takim samym skrócie
2. Odporność na kolizje konkretnych wiadomości (*target collision-resistance*, *preimage resistance*) pierwszego i drugiego rzędu – brak praktycznej możliwości wygenerowania wiadomości o takim samym skrócie jak wskazana wiadomość
3. Jednokierunkowość (*one-wayness*) – brak możliwości wnioskowania o wiadomości wejściowej na podstawie wartości skrótu. Zmiana dowolnego pojedynczego bitu wiadomości powinna zmieniać średnio połowę bitów skrótu w sposób, który nie jest istotnie podatny na kryptoanalizę różnicową.

Poza wymienionymi w niektórych zastosowaniach od funkcji skrótu wymaga się także:

- pseudolosowości (*pseudorandomness*),
- uwierzytelnienia wiadomości (*message authentication*),
- niemożności odróżnienia od tzw. losowej *wyroczni* (*indifferentiability from random oracles*) – uczynienie niemożliwym do znalezienia dla przeciwnika dwóch wiadomości, dla których wartości funkcji skrótu nieznacznie się różnią.

Ponadto nie powinno być możliwości wywnioskowania żadnych użytecznych informacji na temat wiadomości, mając do dyspozycji tylko jej skrót. Dlatego, funkcje haszujące powinny zachowywać się jak funkcje losowe na ile to możliwe, będąc jednocześnie funkcjami deterministycznymi, łatwymi do obliczenia.

Uznanie funkcji za bezpieczną do zastosowań kryptograficznych opiera się zawsze wyłącznie na domniemanej odporności na znane ataki kryptoanalityczne, nie zaś na matematycznych dowodach gwarantujących niemożność złamania. W szczególności bezpieczna funkcja skrótu musiałaby być funkcją jednokierunkową, a istnienie takich funkcji nie zostało dotychczas dowiedzione. Poważne słabości znaleziono w wielu funkcjach skrótu, które historycznie uchodziły za bezpieczne – m.in. w MD2, MD4, SHA, MD5.

Niekiedy zamiast wyspecjalizowanych funkcji skrótu do realizacji podobnych funkcji bezpieczeństwa stosuje się algorytmy zbudowane na bazie szyfrów blokowych – zwłaszcza w kodach uwierzytelniania wiadomości.

Zastosowania bezpiecznych funkcji skrótu

- Weryfikacja integralności plików bądź wiadomości

Istotnym zastosowaniem funkcji skrótu jest weryfikacja spójności danych. Porównanie skrótów dwóch plików umożliwia stwierdzenie, czy w pliku zostały dokonane jakiekolwiek zmiany.

Z tego powodu, większość algorytmów podpisu cyfrowego działa na zasadzie wygenerowania skrótu wiadomości, dzięki czemu można w dowolnym momencie sprawdzić, czy wiadomość jest autentyczna.

Funkcje skrótu używane są również do weryfikacji haseł. W celu zwiększenia bezpieczeństwa, w bazie danych przechowuje się skrót hasła, zamiast tekstu jawnego. Hasło wprowadzone przez użytkownika jest haszowane i dopiero wtedy porównywane ze skrótem w bazie danych. Taki sposób przechowywania haseł utrudnia ich odzyskanie, ze względu na jednokierunkowość funkcji haszujących.

- Identyfikacja plików lub danych

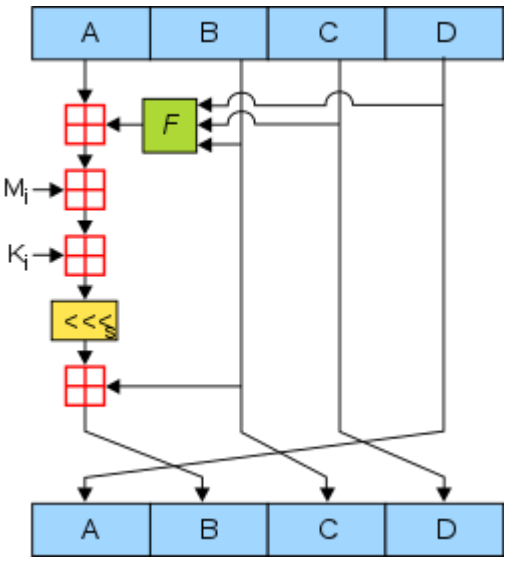
Skrót może również służyć do identyfikacji plików: Systemy kontroli wersji, takie jak *Git* lub *Mercurial* używają funkcji skrótu SHA do identyfikacji różnego rodzaju zawartości (zawartość plików, drzewa katalogów).

Innym zastosowaniem jest używanie skrótów w tablicach z haszowaniem w celu szybkiego wyszukiwania danych.

3. Algorytm MD5 - jest następujący:

1. Doklejenie do wiadomości wejściowej bitu o wartości 1
2. Doklejenie takiej ilości zer, by ciąg składał się z 512-bitowych bloków i ostatniego niepełnego – 448-bitowego
3. Doklejenie 64-bitowego (zaczynając od najmniej znaczącego bitu) licznika oznaczającego rozmiar wiadomości; w ten sposób otrzymywana wiadomość złożona jest z 512-bitowych fragmentów
4. Ustawienie stanu początkowego na 0123456789abcdeffedcba9876543210
5. Uruchomienie na każdym bloku funkcji zmieniającej stan (istnieje przynajmniej jeden blok nawet dla pustego wejścia)
6. Zwrócenie stanu po przetworzeniu ostatniego bloku jako obliczony skrót wiadomości

Funkcja zmiany stanu ma 4 cykle (64 kroki). Stan jest traktowany jako 4 liczby 32-bitowe. W każdym kroku do jednej z tych liczb dodawany jest jeden z 16 32-bitowych fragmentów bloku wejściowego, pewna stała zależna od numeru kroku oraz pewna prosta funkcja boolowska 3 pozostałych liczb. Następnie liczba ta jest obracana (przesuwana cyklicznie z najstarszymi bitami wsuwanymi w najmłodsze pozycje) o liczbę bitów zależną od kroku, oraz jest dodawana do niej jedna z pozostałych liczb.



Funkcje te to:

- W krokach 1 do 16 (cykl 1) funkcja `F(x,y,z) = (x and y) or (neg x and z)`
  $F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
  (jeśli x to y, w przeciwnym wypadku z)
- W krokach 17 do 32 (cykl 2) funkcja `G(x,y,z) = (x and z) or (y and neg z)`
  $G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$
  (jeśli z to x, w przeciwnym wypadku y)
- W krokach 33 do 48 (cykl 3) funkcja `H(x,y,z) = (x xor y xor z)`
  $H(X,Y,Z) = X \oplus Y \oplus Z$
  (suma argumentów modulo 2, lub innymi słowy: czy występuje nieparzysta liczba jedynek w argumentach)
- W krokach 49 do 64 (cykl 4) funkcja `I(x,y,z) = (y xor (x or neg z))`
  $I(X,Y,Z) = Y \oplus (X \vee \neg Z)$
  (jeżeli ($z = 1$ i $x = 0$) wtedy y, w przeciwnym wypadku nie y)

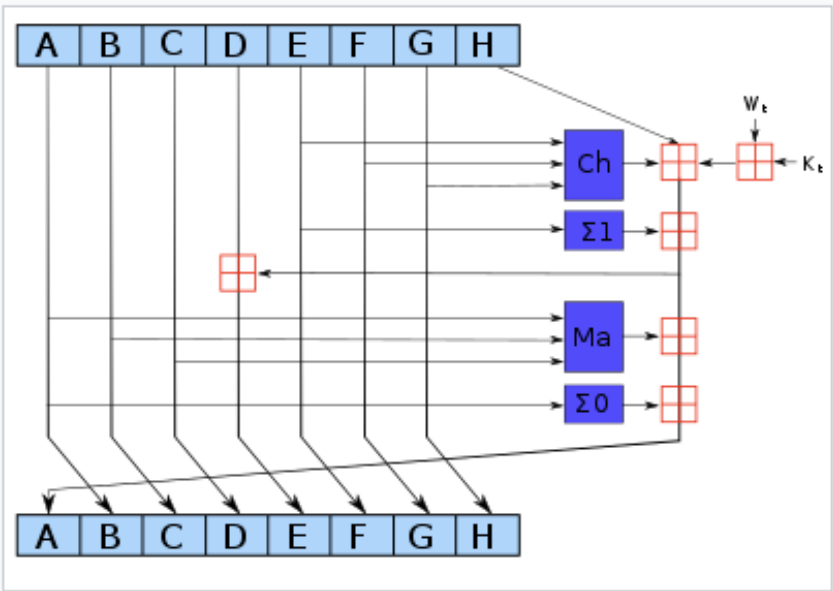
Aby otrzymać wartość stałej w `i`-tym kroku, należy wziąć 32 najstarsze bity z części ułamkowej wyrażenia $|\sin\ i|$.

Podobną budowę mają funkcje skrótu MD4, SHA0 i SHA-1 – różnią się one jedynie postacią funkcji zmieniającej stan, oraz rozmiarem stanu (160 bitów, czyli 5 32-bitowych rejestrów w SHA i SHA1, wobec 128 w MD4 i MD5).

Skróty 128-bitowe są zbyt krótkie, żeby zabezpieczyć przed generowaniem kolizji w oparciu o atak „urodzinowy”. Z tego powodu do większości zastosowań lepiej jest używać skrótów co najmniej 160-bitowych.

4. Algorytmy SHA

NIST opublikował cztery dodatkowe funkcje skrótu z rodziny SHA, nazwane według ich długości skrótu (w bitach): SHA-224, **SHA-256**, SHA-384 i SHA-512. Algorytmy są zbiorczo określane jako SHA-2.



Jedna iteracja w funkcji kompresji rodziny SHA-2. Niebieskie elementy wykonują następujące czynności:
$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$ $\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$
$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$
$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$
Binarna rotacja używa innych stałych dla SHA-512. Podane liczby są dla SHA-256. Czerwony ⊞ jest dodaniem modulo 2^{32}.

SHA-256 i SHA-512 są nowatorskimi funkcjami skrótu liczonymi odpowiednio na 32- i 64-bitowych słowach. Używają różnych ilości przesunięcia i dodatkowych stałych, ale ich struktury są poza tym praktycznie identyczne, różnią się tylko liczbą rund. SHA-224 i SHA-384 są po prostu obciętymi wersjami dwóch pierwszych, obliczone z różnych wartości początkowych.

Porównanie funkcji SHA

W tabeli poniżej *wewnętrzny stan* oznacza „wewnętrzną sumę hash" po każdej kompresji bloku danych.

Algorytm i wariant		Rozmiar wyjścia (bity)	Wewnętrzny rozmiar stanu (bity)	Rozmiar bloku (bitów)	Maksymalny rozmiar wiadomości (bitów)	Rozmiar słowa (bity)	Rundy	Operacje	Znalezione Kolizje	Przykładowa wydajność (MB/s)	
										32b[9]	64b[10]
SHA-0		160	160	512	$2^{64}-1$	32	80	+,and,or,xor,rot	Tak	–	–
SHA-1		160	160	512	$2^{64}-1$	32	80	+,and,or,xor,rot	Tak[11]	153	192
SHA-2	*SHA-256/224*	256/224	256	512	$2^{64}-1$	32	64	+,and,or,xor,shr,rot	Brak	111	139
	SHA-512/384	512/384	512	1024	$2^{128}-1$	64	80	+,and,or,xor,shr,rot	Brak	99	154

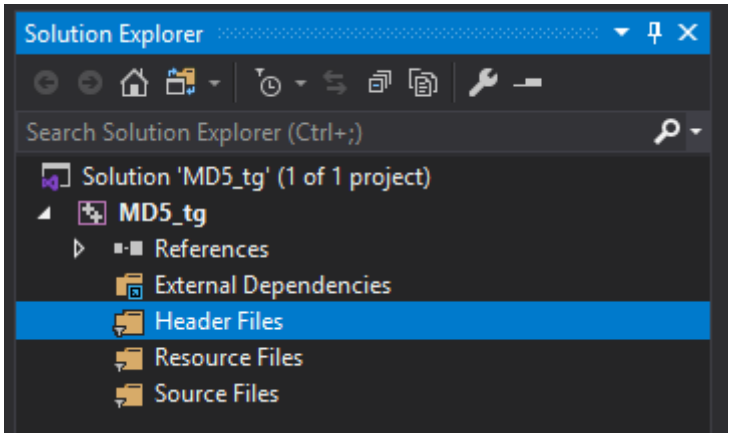
Przedstawiona w tabeli wydajność służy do celów porównawczych. Testy przeprowadzano na platformie Intel Core 2 1,83 GHz z systemem Windows Vista w trybie 32-bitowym (dla wersji 32-bitowej) oraz AMD Opteron 8354 2,2 GHz z systemem Linux (dla wersji 64-bitowej).

Zadania do wykonania i sprawozdanie.

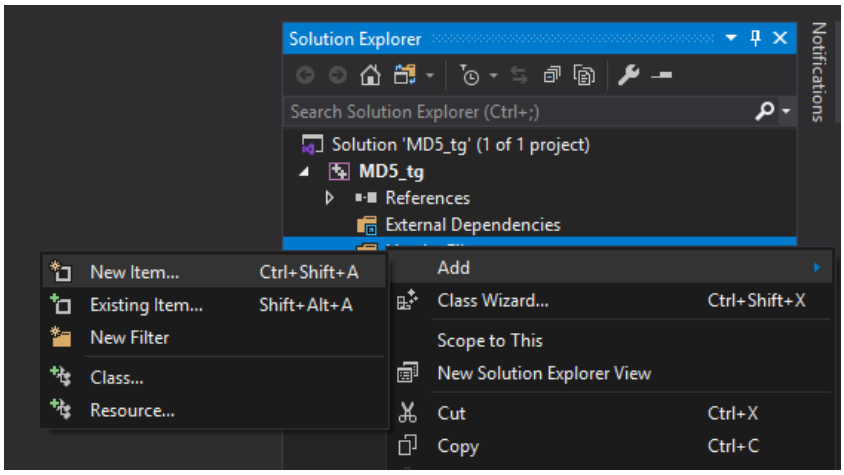
Wykonanie ćwiczenia wymaga środowiska Visual Studio np. w wersji Community.
Instrukcja instalacji znajduje się w pliku *InstalacjaVSComm.pdf* – dostępnym w zestawie do pobrania w ćwiczeniu nr 2.

Ponieważ testować będziemy dwa algorytmy to utworzymy kolejno dwa projekty, dla każdego algorytmu oddzielny, wg następujących kroków:

Po uruchomieniu VS należy wybrać utworzenie pustego projektu (Empty Project) w C++.
Po nadaniu swojej nazwy (np. MD5) i uruchomieniu się tego projektu należy utworzyć (dodać) do niego pliki (ich treści znajdują się w spakowanym, dołączonym, archiwum w katalogach \MD5 oraz \SHA256).
W okienku po prawej (Solution Explorer) jest następujące Menu:



Np. dla pierwszej części tj. algorytmu MD5 trzeba dodać:



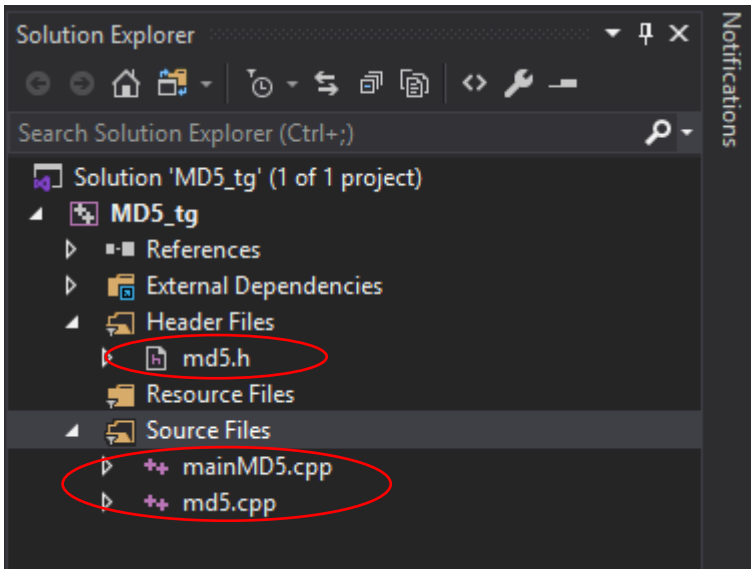
- plik nagłówkowy, z rozszerzeniem *.h* (Header Files->New item->typ: Header File (.h)). (można zmienić jego nazwę (prawy click -> Rename na pliku) na taką jak ma dołączony plik, czyli *md5.h* – i wkleić jego zawartość we właściwe otwarte dla niego okno). Poniższy obrazek pokazuje, że można to również zrobić skrótem Ctrl-Shift-A.

- pliki: źródłowy, z rozszerzeniem *.cpp* (Source Files->New item->typ: C++ File (.cpp))

(zmiana nazwy na np. mainMD5.cpp (prawy click -> Rename na pliku) i wkleić zawartość pliku *main.cpp*;

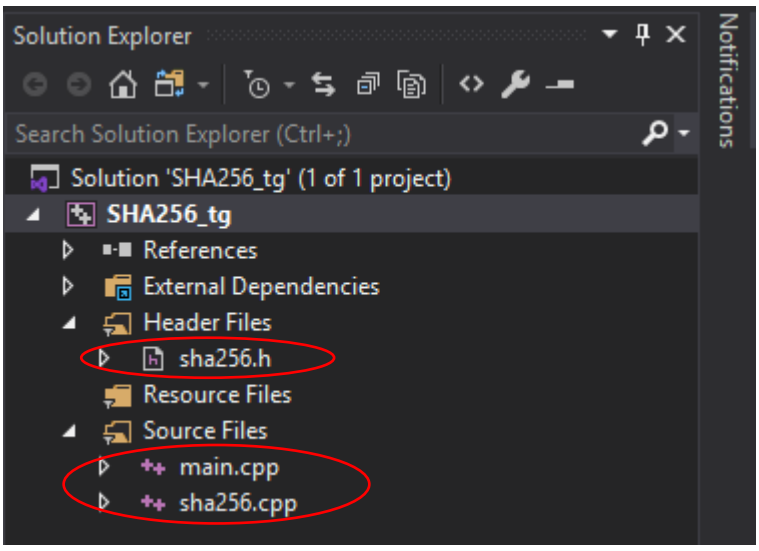
- dodać drugi źródłowy plik, z rozszerzeniem *.cpp* (Source Files->New item->typ: C++ File (.cpp)), (zmiana nazwy na np. *md5.cpp* (prawy click -> Rename na pliku) i wkleić zawartość pliku *md5.cpp*.

Po wykonaniu tych operacji okienko Solution Explorera wygląda następująco:

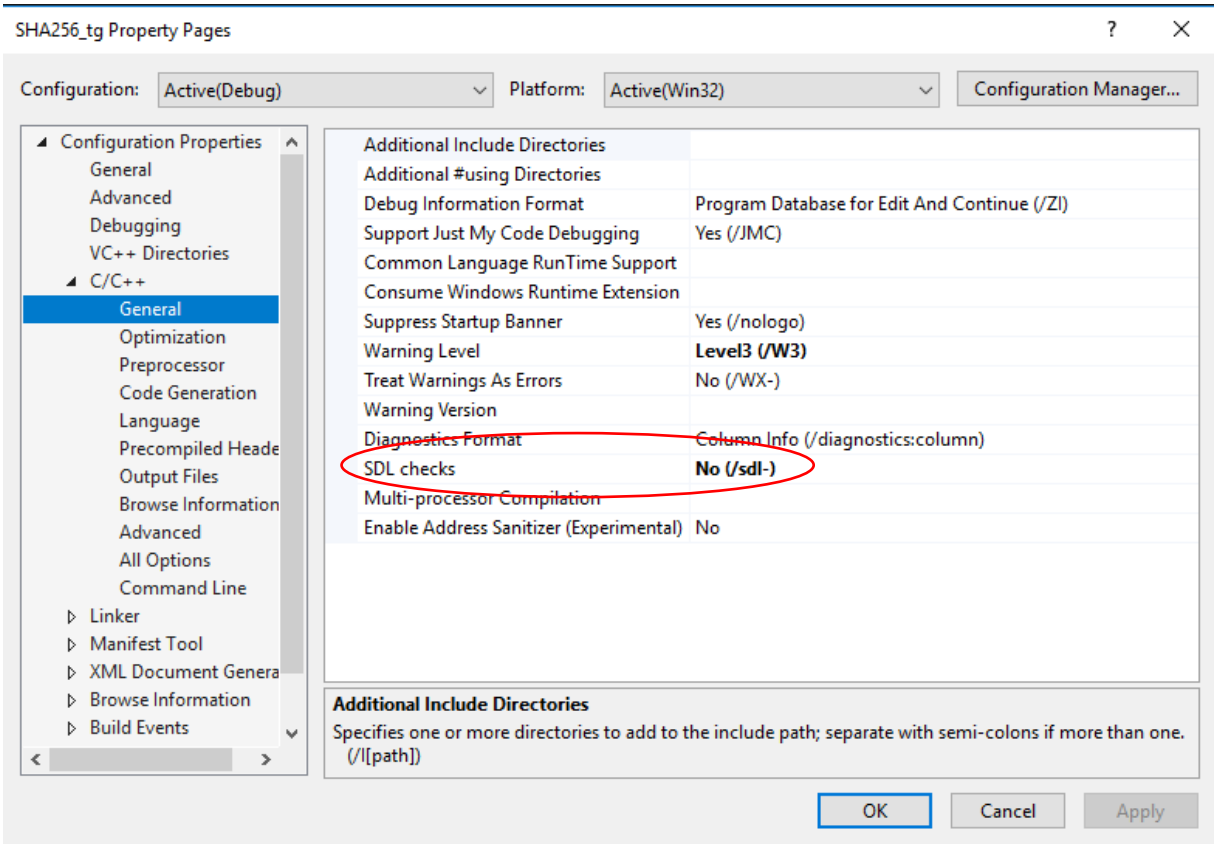


Dla części drugiej ćwiczenia tj. badania algorytmu SHA256 postępujemy analogicznie, z tym, że nadajemy inne nazwy zarówno projektowi, jak i tworzonym plikom ze źródłami.

Rezultat (u mnie) wygląda tak:



Następnie: Uruchomienie debuggera, kompilacja i uruchomienie programu: klawiszem **F5**.
UWAGA: Może pojawić się błąd podczas kompilacji, że jakaś operacja jest uznawana za niebezpieczną. Wówczas należy otworzyć własności projektu (prawy click na Nazwie Projektu->Properties-> odszukać w C/C++->General-> SDL checks i zmienić na **No**).
Ilustracja poniżej powinna być pomocna.



Uruchamiamy program klawiszem **F5**.

Jeśli wszystko przebiegnie pomyślnie otworzy się okienko z prostym **Menu**.
Wybór mamy z trzech opcji:

1. Obliczenie funkcji skrótu dla ciągu wpisanego wprost z klawiatury
2. Obliczenie funkcji skrótu dla pliku wczytywanego z zewnątrz (np. dołączony plik *test.txt*, który powinien znaleźć się w katalogu Projektu z pozostałymi źródłami (trzeba ich szukać w katalogu macierzystym użytkownika: \source\repos\NazwaProjektu\ ).
3. Wyjście z programu.

Program ćwiczenia obejmuje przeprowadzenie testów obliczania wartości badanych funkcji skrótu (nazwanych tutaj: *fingerprints* – odciski palca) dla różnych przypadków ciągów wejściowych.
W szczególności:
- Proszę zwrócić uwagę na to jak zmieni się *fingerprint* jeśli w ciągu wejściowym zmienimy tylko jedną literę np. dla zdania: „Ala ma czarnego kota” i „Ola ma czarnego kota”. Czy wyniki będą podobne czy różnią się znacznie?
- Proszę przetestować różnej długości ciągi wejściowe – jak zmienia się długość „odcisku palca”?

Otrzymane rezultaty opisać w sprawozdaniu, zamieszczając wybrane ekrany z wynikami (np. używając narzędzia Windows: *snipping-tool*).

Nazwa wysyłanego pliku ze sprawozdaniem powinna odpowiadać wzorcowi:

[BKE_NS]_Labor04_Imie_Nazwisko.pdf

Zadanie nieobowiązkowe – dla zaawansowanych:

Programy mają bardzo ubogie (żeby nie powiedzieć prymitywne) interfejsy użytkownika.
Może ktoś z Państwa zechciałby stworzyć bardziej atrakcyjne wizualnie środowisko graficzne do jego prezentacji?