Jeżeli globalna zmienna całkowita nie została zainicjalizowana, to ma przypadkową wartość początkową.

Jeżeli globalna zmienna logiczna nie została zainicjalizowana to jej wartością początkową jest na pewno fałsz.  
Jeżeli lokalna zmienna rzeczywista nie została zainicjalizowana, to ma przypadkową wartość początkową.

Które z poniższych wycinków kodu wypisują liczbę 2?

Int i = 0;

I++ || ++i; std::cout << i; NIE -> TAK (wytestowany kod zwrócił 2)

(i = 1) || (i = 2); std::cout << i;

(i = 1) && (i = 2); std::cout << i;

Elementy wektora są indeksowane od zera.

Metoda size zwraca indeks hipotetycznego elementu za ostatnim elementem wektora. NIE -> TAK  
(jeżeli wektor ma indeksy np. 0, 1, 2, 3, 4, to size zwróci 5, czyli HIPOTETYCZNY indeks elementu za ostatnim elementem wektora)

Odwołanie do nieistniejącego elementu wektora zawsze jest naruszeniem ochrony pamięci.

Można spowodować, by program czytający dane ze strumienia typu std::ifstream przeczytał je w rzeczywistości z klawiatury.  
Standardowe wejście to po prosty klawiatura.  
Można spowodować, by program czytający dane ze strumienia std::cin przeczytał je w rzeczywistości z pliku dyskowego.

Które z poniższych funkcji powodują błędy kompilacji?

Int &function(int &r) {return ++r;}

Int &function(int &r) {return r++;}

Int function(int &r) {return r++;}

Typ każdej zmiennej musi być podany przez programistę.

Typ zmiennej może się zmieniać podczas wykonania programu.

Typ każdej zmiennej musi być znany podczas kompilacji.

Argumentem operatora sizeof może być stała dosłowna.

Argumentem operatora sizeof może być nazwa zmiennej.

Argumentem operatora sizeof może być nazwa typu.

Jeżeli funkcja zwraca wynik, to po instrukcji return może stać zmienna, stała lub obiekt tymczasowy bez nazwy.

Jeżeli funkcja zwraca wynik typu double, to po instrukcji return musi stać wartość typu double.

Jeżeli w funkcji main nie ma instrukcji return, to funkcja ta nie zwraca żadnej wartości.

W pewnych przypadkach manipulator std::setprecision ustawia minimalną liczbę cyfr znaczących.

Manipulator std:: setw wpływa tylko na najbliższy wydruk.

W pewnych przypadkach manipulator std::setprecision ustawia dokładną liczbę cyfr po przecinku.

Które z poniższych definicji tworzą wektor o elementach 2 i 7?  
std::vector<int> vector {2,7};

std::vector<int> vector (2,7);

std::vector<int> vector = {2,7};

Wartość typu char to liczba całkowita.

Jeżeli zmienna typu int zawiera kod ASCII znaku, to jest domyślnie drukowana na standardowe wyjście jako znak.

Char to typ ze znakiem, więc istnieją znaki o ujemnych kodach ASCII. TAK -> NIE (traktuj kod ascii jak id znaku – nie ma ujemnych id)

Jeżeli uruchamiając program w linii poleceń za nazwą egzakutabli napisano „Ala ma kota”, to przekazany do funkcji main argument argc ma wartość dwa. NIE -> TAK

„Ala ma kota” traktowane jest jako jeden ciąg znaków/jeden obiekt/jedna zmienna.   
Argc liczy się tak: program.exe zmienna1 zmienna2 „tekst tekst tekst”  
 1 2 3 4

To tylko poglądowo – suma argumentów argc jest 4, natomiast indeksuje się od 0: 0, 1, 2, 3

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Jeżeli uruchamiając program w linii poleceń za nazwą egzakutabli napisano Ala ma kota, to przekazany do funkcji main argument argc ma wartość trzy. TAK -> NIE   
W argc wlicza się również nazwę programu

Jeżeli uruchamiając program w linii poleceń podano tylko nazwę egzakutabli, to przekazany do funkcji main argument argc ma wartość jeden.

Wynikiem operatora preinkrementacji jest referencja niemodyfikująca.

Wynikiem operatora preinkrementacji jest obiekt tymczasowy bez nazwy.

Wynikiem operatora preinkrementacji jest referencja modyfikująca.

Jeżeli umieszczona w ciele pętli definicja zmiennej typu wbudowanego nie jest połączona z inicjalizacją, to zmienna ta ma przypadkową wartość początkową.

Jeżeli umieszczona w ciele pętli definicja jest połączona to inicjalizacja ta wykonuje się tylko w pierwszej iteracji.

Jeżeli umieszczona w ciele pętli definicja zmiennej jest połączona z inicjalizacją, inicjalizacja ta wykonuje się w każdej iteracji.

Porównanie dwóch łańcuchów typu std::string operatorem == daje w wyniku prawdę jedynie jeśli łańcuchy są identyczne.

Porównanie dwóch łańcuchów typu std::string operatorem < daje w wyniku prawdę jedynie jeśli pierwszy łańcuch wypada wcześniej w kolejności leksykograficznej.

Porównanie dwóch łańcuchów typu std::string operatorem < daje w wyniku prawdę jedynie jeśli pierwszy łańcuch jest krótszy od drugiego.

Które z poniższych wyrażeń zwracają wartość logiczna prawda?

True && !false

!(true && false)

!true && false

Które z poniższych wyrażeń zwracają liczby pseudolosowe od -1 do 7 włącznie?

Std::rand() % 9 - 1

Std::rand() % 7 – 1

7- std::rand() % 9

Które z poniższych wyrażeń zawracają wynik typu char?

Int(`c`) – `a`

`a` + 3

Char(100) - `a`

Wynik wyrażenia można pozostawić niewykorzystany.

Wynikiem wyrażenia jest zawsze obiekt tymczasowy bez nazwy.

Wyrażenie tym się różni od instrukcji, że zwraca wynik.

Unsigned i unsigned int to dwa różne typy.

Short I signed short int to dwa różne typy.

Int i signed int to dwa różne typy.

Które z poniższych wywoła funkcji powodują błędy w kompilacji?

Void function(int a, int b = 0, int c=0) {}

Void function(int a, int b) {}

Function(1);

Function(1,2,3); NIE -> TAK (nie wywoła się druga funkcja, bo za dużo argumentów podano)

Function(1,2); TAK -> NIE (wywoła się, bo druga funkcja przyjmnie obie wartości, a pierwsza automatycznie da brakującą wartość c)

Można utworzyć modyfikującą referencję obiektu tymczasowego bez nazwy.

Obiekt tymczasowy bez nazwy jest niemodyfikowalny. TAK -> NIE (choć tu akurat nie jestem pewny)

Utworzenie niemodyfikującej referencji obiektu tymczasowego w pewnych przypadkach przedłuża istnienie tego obiektu.

Kilka argumentów funkcji może mieć jednakową wartość domyślną

Argumenty z wartościami domyślnymi muszą znajdować na końcu listy argumentów.

Jeden argument funkcji może mieć kilka wartości domyślnych.

Wynikiem operatora dzielenia jest referencja modyfikująca.

Wynikiem operatora dzielenia jest referencja niemodyfikująca.

Wynikiem operatora dzielenia jest obiekt tymczasowy bez nazwy.

Dla których definicji wektora poniższa linia drukuję cyfrę 1?

Std :: cout << vector [1][1];

Std::vector<std::vector<int>> vector(3, std::vector<int> {2,1});

Std::vector<std::string> vector(3,std::string{2,1});

Std::vector<std::vector<char>> vector(3,std::vector<char> {2,1});

Zmienna utworzona w ciele pętli do while jest widoczna w warunku tej pętli.

Zmienna utworzona przed pierwszym średnikiem w okrągłym nawiasie pętli for jest widoczna w warunku tej pętli;

Zmienna utworzona w ciele pętli while jest widoczna w warunku tej pętli.

Istnieje niejawna konwersja stałej napisowej do typu std::string.

Istnieje konwersja typu std::string do typu std::vector<char>. TAK -> NIE  
CHYBA nie ma takiej konwersji, to znaczy da się zaprogramować coś takiego pętlą, ale jako tako konwersji nie ma

Istnieje konwersja stałej napisowej do typu std::vector<char>.

Które z poniższych wycinków kodu dają wydruk 123?

Int i = 1; while (i<=3) {std::cout << i; ++i;}

For (int i =1; i <=3;i++) {std::cout<<i;}

Int i=1; do {std::cout << i; ++i;} while(i<=3)