1. Co wypisuje poniższy kod?
std::vector<int> vector {6, 3, 5, 7, 1, 9, 2, 4, 8};
auto result = *std::find(vector.begin() + 2, vector.end() - 2, 3);
std::cout << result << std::endl;

4
3
Co innego

2. Które z poniższych linii drukują napis alamakota?
std::string s1 = "ala", s2 = "ma", s3 = "kota";

std::cout << (s1 += s2 += s3) << std::endl;
std::cout << (s1 + s2 += s3) << std::endl;
std::cout << (s1 += s2 + s3) << std::endl;

3. Zadanie:
Jeżeli trzecim argumentem wywołania algorytmu std::for_each jest funkcja lub wyrażenie lambda przyjmujące referencję modyfikującą, to pierwszymi dwoma argumentami muszą być iteratory modyfikujące
Jeżeli pierwszymi dwoma argumentami wywołania algorytmu std::for_each są iteratory modyfikujące, to trzecim musi być funkcja lub wyrażenie lambda przyjmujące referencję modyfikującą.
Jeżeli trzecim argumentem wywołania algorytmu std::for_each jest funkcja lub wyrażenie lambda przyjmujące referencję niemodyfikującą, to pierwszymi dwoma argumentami muszą być iteratory niemodyfikujące.

4. Zadanie:
Wynikiem operatora dzielenia jest referencja niemodyfikująca.
Wynikiem operatora dzielenia jest referencja modyfikująca.
Wynikiem operatora dzielenia jest obiekt tymczasowy bez nazwy.

5. Zadanie:
Istnieje niejawna konwersja stałej napisowej do typu std::string
Istnieje konwersja stałej napisowej do typu std::vector<char>
Istnieje konwersja typu std::string do typu std::vector<char>

6. Które z poniższych wycinków kodu powodują błędy kompilacji?
int i, j; [] {i + j; }
const int i, j; [i, j] {i + j; }
int i, j; [=] {i += j; }

7. Które z poniższych wycinków kodu wypisują przy pierwszym wykonaniu liczbę 1?
int &function(){
** static int variable = 0;**
** return variable; }**

++function(); std::cout << function();
std::cout << (++function())++;
function()++; std::cout << function();

8. Zadanie:
Wyrażenie tym się różni od instrukcji, że zwraca wynik.
Wynik wyrażenia można pozostawić niewykorzystany.
Wynikiem wyrażenia jest zawsze obiekt tymczasowy bez nazwy.

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

10. Które z poniższych wywołań funkcji powodują błędy kompilacji?
void function(int a, int b = 0, int c = 0) {}
void function(int a, int b) {}

function(1, 2, 3);
function(1, 2);
function(1);

11. Które z poniższych linii powodują błędy kompilacji?
int i;

++i = 1;
(i = 1)++;
i++ = 1;

12. Dla języka C++ odpowiedz na pytania prawda czy fałsz:
Na podstawie indeksu można wyznaczyć iterator elementu wektora.
Istnieje konwersja indeksu do iteratora elementu w wektorze.
Istnieje niejawna konwersja iteratora modyfikującego do niemodyfikującego.

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

14. Zadanie:
Różnica między iteratorem elementu a iteratorem początkowym wektora to indeks tego elementu.
Różnica między iteratorami dwóch elementów wektora jest równa różnicy między ich indeksami.
Różnica między końcowym a początkowym iteratorem wektora to indeks hipotetycznego elementu za ostatnim elementem wektora.

15. Zadanie:
Jeden argument funkcji może mieć kilka wartości domyślnych.
Argumenty z wartościami domyślnymi muszą się znajdować na końcu listy argumentów.
Kilka argumentów funkcji może mieć jednakową wartość domyślną

16. Zadanie:
short i signed short int to dwa różne typy.
unsinged i unsinged int to dwa różne typy.
int i signed int to dwa różne typy.

17. Zadanie:
Obiekt tymczasowy bez nazwy jest niemodyfikowalny.
Można utworzyć modyfikującą referencje obiektu tymczasowego bez nazwy.
Utworzenie niemodyfikującej referencji obiektu tymczasowego w pewnych przypadkach przedłuża istnienie tego obiektu.

18. Zadanie:
Wynikiem operatora preinkrementacji jest referencja niemodyfikująca.
Wynikiem operatora preinkrementacji jest obiekt tymczasowy bez nazwy.
Wynikiem operatora preinkrementacji jest referencja modyfikująca.

19. Zadanie:
Jeżeli umieszczona w ciele pętli definicja zmiennej jest połączona z Inicjalizacja to inicjalizacja ta wykonuje się tylko w pierwszej iteracji.
Jeżeli umieszczona w ciele pętli definicja zmiennej typu wbudowanego nie jest połączony z inicjalizacja to zmienna ta ma przypadkową wartość początkową.
Jeżeli umieszczona w ciele pętli definicja zmiennej jest połączone z Inicjalizacja to Inicjalizacja ta wykonuje się w każdej pętli.

20. Zadanie:
Typ każdej zmiennej musi być podany przez programistę.
Typ zmiennej może się zmieniać podczas wykonania programu.
Typ każdej zmiennej musi być znany podczas kompilacji.

21. Zadanie:
Jeden argument funkcji może mieć kilka wartości domyślnych.
Kilka argumentów funkcji może mieć jednakową wartość domyślną.
Argumenty z wartościami domyślnymi muszą się znajdować na końcu listy argumentów.

22. Co wypisuje poniższy wycinek kodu:

```
std::vector<int> vector {1,2};
for (int i=1; i < vector.size(); i *= 2) {
        vector.push_back(vector[i]); }
for (int i=1; i < vector.size(); i *= 2) {
        std::cout << vector[i]; }
```

dwie cyfry
cztery cyfry
trzy cyfry

23. Zadanie:
W pewnych przypadkach manipulator std::setprecision ustawia dokładną liczbę cyfr po przecinku.
Manipulator std::setw wpływa tylko na najbliższy wydruk.
W pewnych przypadkach manipulator std::setprecision ustawia minimalną liczbę cyfr znaczących.

24. Zadanie:
Wartość typu char to liczba całkowita.
char to typ ze znakiem, więc istnieją znaki o ujemnych kodach ASCII.
Jeżeli zmienna typu int zawiera kod ASCII znaku, to jest domyślnie drukowana na standardowe wyjście jako znak.

25. Zadanie:
Funkcja przyjmująca jako argument inną funkcje lub wyrażenie lambda to przykład funkcji wyższego rzędu.
Jeżeli argument funkcji jest zadeklarowany przy pomocy szablonu std::function, to można jako ten argument przekazać zwykłą funkcję lub wyrażenie lambda.
Argumentami funkcji wyższego rzędu mogą być tylko takie funkcje lub wyrażenia lambda, które zwracają wynik.

26. Zadanie:
Argumentem operatora sizeof może być nazwa typu.
Argumentem operatora sizeof może być stała dosłowna.
Argumentem operatora sizeof może być nazwa zmiennej.

27. Zadanie:
Dwuargumentowa funkcja std::getline wczytuje znak końca linii, ale nie dołącza go do wczytanego łańcucha
Dwuargumentowa funkcja std::getline wczytuje znak końca linii i dołącza go do wczytanego łańcucha.
Trzeci argument funkcji std::getline określa, czy znak końca linii ma zostać dołączony do wczytanego łańcucha.

28. Zadanie:
Jeżeli funkcja zwraca wynik, to po instrukcji return można stać zmienna stała lub obiekt tymczasowy bez nazwy.
Jeżeli funkcja main nie ma instrukcji return, to funkcja ta nie zwraca żadnej wartości.
Jeżeli funkcja zwraca wyniku typu double, to po instrukcji return musi stać wartość typu double.

29. Zadanie:
Wszelkie operacje na referencji są równoznaczne z operacjami na obiekcie, do którego się odnosi.
Ta sama zmienna referencyjna może się odnosić raz do jednego obiektu, a za chwilę do drugiego.
Utworzenie zmiennej referencyjnej musi być połączone z inicjalizacją.

30. Zadanie:
Jeżeli uruchamiając program w linii poleceń za nazwą egzekutabli napisano „ala ma kota”, to przekazany do funkcji main argument argc ma wartość dwa.
Jeżeli uruchamiając program w linii poleceń za nazwą egzekutabli napisano „ala ma kota”, to przekazany do funkcji main argument argc ma wartość trzy.
Jeżeli uruchamiając program w linii polecań podano tylko nazwę egzekutabli, to przekazany do funkcji main argument argc ma wartość jeden

31. Zadanie:
Operatorem >> można odczytać znak biały,
Operator >> opuszcza wszystkie znaki białe poprzedzające odczytywaną wartość,
Wciśnięcie Enter zawsze kończy odczyt danych operatorem >>.

32. Dla których definicji wektora poniższa linia drukuje cyfrę 1?
std::vector<std::vector<int>> vector(3, std::vector<int> {2, 1});
std::vector<std::vector<char>> vector(3, std::vector<char> {2, 1});
std::vector<std::string> vector(3, std::string {2, 1});

33. Które z poniższych linii drukują nierozdzielone cyfry?
std::cout << std::setw(2) << std::right << 1 << 2 << std::endl;
std::cout << std::setw(2) << 1 << std::right << 2 << std::endl;
std::cout << std::right << 1 << std::setw(2) << 2 << std::endl;

34. Zadanie:
Konwersja typu char do typu bool musi być jawna.
Liczby ujemne konwertują się do wartości logicznej fałsz
Istnieje niejawna konwersja typu int do typu bool.

35. Zadanie:
W programie mogą istnieć funkcje o tej samej nazwie i argumentach, ale różniące się typem wyniku.
Funkcje przeciążone to funkcje o tej samej nazwie, ale różniące się liczbą lub typem argumentów.
Wszystkie funkcje o tej samej nazwie muszą zwracać wynik tego samego typu.

36. Zadanie:
Liczba 3.21 ma trzy cyfry znaczące.
Liczba 3.2e05 ma pięć cyfr znaczących.
Liczba 3.210 ma cztery cyfry znaczące.

37. 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);

38. Zadanie:
Każde wyrażenie lambda ma inny typ
Zamiast używać słowa kluczowego auto, w definicji nazwanego wyrażenia lambda można samemu podać jego typ
Typ wyrażenia lambda opisany szablonem std::function

39. Zadanie:
Wyłuskanie iteratora końcowego wektora zawsze jest błędem.
Wyłuskanie iteratora początkowego wektora nigdy nie jest błędem.
Jeżeli iterator początkowy wycinka jest równy jego iteratorowi końcowemu, to wyłuskanie tych iteratorów zawsze jest błędem

40. Zadanie:
W kodzie ASCII istnieje specjalny znak końca pliku i każdy plik tekstowy kończy się tym znakiem.
W konsoli systemu windows koniec pliku symuluje się wciskając Ctrl-Z, a potem Enter
W terminalu systemu Linux koniec pliku symuluje się wciskając Ctrl-C.

41. Które z poniższych wycinków kodu daję wydruk 123?
for (int i = 1; i <= 3; ++i) {std::cout << i; }
int i = 1; do {std::cout << i; ++i; } while(i <= 3);
int i = 1; while (i <= 3) {std::cout << i; ++i; }

42. Zadanie:
Jeżeli warunek pętli for nie jest spełniony, to zamiast wykonania ciała pętli obliczane jest wyrażenie za drugim średnikiem w okrągłym nawiasie pętli, po czym pętla się kończy.
Jeżeli pomiędzy pierwszym a drugim średnikiem w okrągłym nawiasie pętli for nie umieszczono żadnego wrażenia, to nastąpi błąd kompilacji.
Warunek pętli for jest sprawdzany bezpośrednio po wykonaniu ciała pętli.

43. Które z poniższych wycinków kodu wypisują liczbę 2?
int i = 0;
(i = 1) && (i = 2); std::cout << i;
(i = 1) || (i = 2); std::cout << i;
i++ || ++i; std::cout << i;

44. Zadanie:
Odczyt z pliku nie zmienia jego zawartości. Zawartość pliku można więc odczytać za pomocą funkcji przyjmującej skojarzony z tym plikiem strumień przez referencję nie modyfikującą.
Funkcja przyjmująca strumień typu std::ifstream przez wartość nie kompiluje się.
Zawartość pliku można odczytać za pomocą funkcji, która przyjmuje skojarzony z tym plikiem strumień przez wartość.

45. Które z poniższych linii powodują błędy kompilacji?
(int &e) {e++; }(0);
(int e) {e++; }(0);
[&e] {e++; } (0);

46. Zadanie:
Pusty łańcuch typu std::string to taki, który zawiera tylko znaki białe.
W odniesieniu do łańcucha typu std::string metoda size może dać inny wynik niż operator sizeof.
Rozmiar łańcucha typu std::string to liczba znaków z wyłączeniem znaków białych.

47. Zadanie:
Iteratora niemodyfikujacego nigdy nie można przesunąć na nastepny element.
Za pośrednictwem iteratora stałego można w pewnych przypadkach zmodyfikować element.
Iteratora stałego nigdy nie można przesunąć na następny element.

48. Zadanie:
Jeśli funkcja zwraca wynik przez wartość, to wynikiem wywołania tej funkcji jest obiekt tymczasowy bez nazwy.
Jeśli funkcja przyjmuje argument przez wartość, to podczas wywołania funkcji tworzona jest kopia argumentu.
Przez wartość można przekazać do funkcji zarówno zmienną, stałą, jak i obiekt tymczasowy bez nazwy.

49. Zadanie:
Obliczane są zawsze oba operandy alternatywy i koniunkcji logicznej.
Jeżeli pierwszy operand koniunkcji logicznej jest prawdą, to drugi nie jest obliczany.
Jeżeli pierwszy operand alternatywy logicznej jest prawdą, to drugi nie jest obliczany.

50. Zadanie:
Istnieje niejawna konwersja referencji jednego typu do referencji innego typu.
Istnieje niejawna konwersja referencji niemodyfikujące do modyfikującej.
Istnieje niejawna konwersja referencji modyfikującej do niemodyfikujące.

51. Zadanie:
Od iteratora modyfikujacego mozna odjąć niemodyfikujacy.
Jeśli iterator modyfikujacy i niemodyfikujacy wskazuja na ten sam element wektora, to ich porównanie operatorem == daje w wyniku prawdę.
Porównanie dwóch iteratorów operatorem > daje w wyniku prawdę jedynie jeśli iterator po lewej stronie operatora wskazuje na element o większym ind…

52. Zadanie:
Wynikiem operatora *= jest referencja modyfikująca.
Wynikiem operatora *= jest obiekt tymczasowy bez nazwy.
Wynikiem operatora *= jest referencja niemodyfikujaca.

53. Które z poniższych pętli poprawnie wypisują wszystkie elementy wektora?
for (auto i = vector.begin(); i < vector.end(); std::cout << *i++);
for (auto i = vector.begin(); i < vector.end(); std::cout << *++i);
for (auto i = vector.begin(); i < vector.end(); std::cout << ++*i);

54. Zadanie:
Istnieje niejawna konwersja elementu tablicy argv do liczby całkowitej.
Aby odczytać liczbę całkowitą podaną jako argument wywołania programu, należy użyć operatora konwersji do liczby całkowitej.
Argumenty wywołania programu są przekazywane do funkcji main jako napisy w stylu języka c.

55. Zadanie:
Jeżeli wyrażenie lambda przechwytuje zmienną lokalną przez wartość, to działa na jej kopii i może tę kopię modyfikować.
Aby wyrażenie lambda mogło korzystać ze zmiennej lokalnej zdefiniowanej w swoim ciele, to musi ją przechwycić
Wyrażenie lambda może przechwycić inne wyrażenie lambda.

56. Zadanie:
Ostatnim elementem łańcucha typu std::string jest bajt zerowy.
Łańcuch typu std:: string może przechowywać tekst w kodowaniu UTF-8.
Elementy łańcucha typu std::string są typu char.

57. Zadanie:
Funkcja przyjmująca argument przez referencje niemodyfikującą działa na kopii
Aby funkcja mogła zmodyfikować zmienną przekazaną jako argument wywołania, musi przyjmować ten argument przez referencję.
Funkcja przyjmująca argument przez wartość działa na kopii argumentu przekazanego w wywołaniu.

58. Dla których definicji wektora poniższy wycinek kodu wypisuje liczbę 5?
int index = 0;
int result = std: :count_if(vector.begin(), vector.end(), [&](int element)
{return index++ == element; });
std::cout << result << std: :endl;
std::vector<int> vector {0, 3, 2, 6, 7, 5, 1, 4, 8, 9};
std::vector<int> vector {5, 1, 8, 3, 4, 7, 6, 2, 0, 9};
std::vector<int> vector {9, 4, 2, 3, 0, 1, 6, 7, 8, 5};

59. Zadanie:
Każda funkcja musi kończyć się instrukcją return.
Instrukcja return może znajdować się w pętli.
W funkcji może znajdować się najwyżej jedna instrukcja return.

60. Zadanie:
Wyrażenie std::rand() / RAND_MAX zwraca liczby całkowite z przedziału od zera do jednego włącznie.
Wyrażenie std::rand() % RAND_MAX zwraca liczby całkowite z przedziału od zera do RAND_MAX włącznie.
Wyrażenie std::rand() zwraca liczby całkowite z przedziału od zera do RAND_MAX włącznie.

61. Zadanie:
Stała liczbowa 7.5F jest typu double.
Stała liczbowa 7 jest typu int.
Stała liczbowa 7. jest typu float.

62. Zadanie:
Wynikiem operatora postinkrementacji jest referencja niemodyfikująca.
Wynikiem operatora postinkrementacji jest referencja modyfikująca.
Wynikiem operatora postinkrementacji jest obiekt tymczasowy bez nazwy.

63. Zadanie:
Pętlę while zawsze można zastąpić pętlą do while.
Pętlę for zawsze można zastąpić pętlą while.
Pętlę do while zawsze można zastąpić pętlą while.

64. 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++; }

65. Co wypisuje poniższy kod:
std::istringstream stream("3.14159");
int integer;
bool result(stream >> integer);
std::cout << result << std::endl;

3
0
1

66. Które z poniższych linii zawierają poprawny składniowo kod w języku C++?
std::printf("Hello");
print('Hello')
System.out.println(„Hello")

67. Zadanie:
Domyślnie wyrażenie lambda może korzystać ze zmiennych lokalnych zdefiniowanych w tym samym zakresie, co wyrażenie.
Wyrażenie lambda może przechwycić zmienną lokalną przez wartość lub referencję.
Wyrażenie lambda może swobodnie korzystać ze zmiennych globalnych.

68. Zadanie:
Odwołanie do nieistniejącego elementu wektora zawsze jest naruszeniem ochrony pamięci
Elementy wektora są indeksowane od zera .
Metoda size zwraca indeks hipotetycznego elementu za ostatnim elementem wektora .

69. Które z poniższych pętli poprawnie zliczają zawarte w pliku znaki niebiałe?
std :: ifstream file (" file.txt ") ; int count = 0 ;
for (char character ; file >> character ; ++ count) ;
for (char character ; file.get (character) ; ++ count) ;
for (char character ; file.good ( ) ; ++ count) { file >> character ; }

70. Zadanie:
Porównanie dwóch łańcuchów typu std :: string operatorem < daje w wyniku prawdę jedynie jeśli pierwszy łańcuch jest krótszy od drugiego .
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 łańcuchy te są identyczne .

71. Które z poniższych wyrażeń zwracają liczby pseudolosowe od -1 do 7 włącznie ?
std :: rand ( ) % 9 - 1
7- std :: rand ( ) % 9
std :: rand ( ) % 7 – 1

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

73. Które z poniższych wyrażeń zwracają wynik typu char ?
int (' c ') - 'a'
' a ' + 3
char (100) - ' a '

74. Które z poniższych wyrażeń zwracają wartość logiczna prawda?
True && !false
!(true && false)
!true && false

75. Zadanie:
Przed pierwszym średnikiem w okrągłym nawiasie pętli for można umieścić dowolne wyrażenie
Przed pierwszym średnikiem w okrągłym nawiasie pętli for można umieścić dowolnie dużo zmiennych
Za drugim średnikiem w okrągłym nawiasie pętli for można umieścić dowolne wyrażenie

76. Zadanie:
Można utworzyć modyfikującą referencje elementu pojemnika typu std::set.
Można utworzyć modyfikującą referencje pojemnika typu std::set.
Element typu std::set można zmodyfikować.

77. Zadanie:
Alias typu można utworzyć zarówno przy pomocy słowa kluczowego typedef, jak i przy pomocy słowa kluczowego using.
Typ i jego alias to dwa różne typy.
Typ std::string to alias typu vector<char>.

78. Zadanie:
Za pośrednictwem iteratora modyfikującego można modyfikować klucze pojemnika typu std::map.
Pojemnik typu std::map nie ma iteratorów modyfikujących.
Za pośrednictwem iteratora modyfikującego można modyfikować wartości typu std::map.

79. Zadanie:
Pojemnik typu std::set<int> ma iteratory typu std::set<int>::iterator ale są one niemodyfikujące. -> (są stałe)
Pojemnik typu std::set nie ma iteratorów modyfikujących.
Pojemnik typu std::set<int> nie ma iteratorów typu std::<int>::iterator.

80. Zadanie:
W pojemniku typu std::map klucze mogą się powtarzać.
W pojemniku typu std::map para o zadanym kluczu i wartości może wystąpić najwyżej raz.
W pojemniku typu std::map wartości mogą się powtarzać.

81. Zadanie:
Słowo kluczowe auto to żądanie by kompilator sam ustalił typ zmiennej.
Auto to jeden z typów wbudowanych jak int, float, etc.
Zmienną można zdefiniować przy pomocy słowa kluczowego auto tylko gdy jest połączona z inicjalizacją.

82. Zadanie:
Można konwertować iteratory do iteratorów wstecznych i na odwrót.
Wynikiem konwersji iteratora zwykłego do wstecznego jest iterator wsteczny wskazujący o jedną komórkę bardziej w lewo niż zwykły.
Wynikiem konwersji iteratora wstecznego do zwykłego jest iterator zwykły wskazujący o jedną komórkę bardziej w prawo niż wsteczny.