Pytania

1. Co przechowuje wskaźnik w C++?

A) Wartość zmiennej
B) Adres w pamięci
C) Typ zmiennej
D) Rozmiar zmiennej

2. Jaki operator służy do pobrania adresu zmiennej?

A) `*`
B) `&`
C) `#`
D) `@`

3. Co robi operator dereferencji `*` użyty na wskaźniku?

A) Zwraca adres wskaźnika
B) Usuwa wskaźnik z pamięci
C) Zwraca wartość przechowywaną pod adresem wskaźnika
D) Inkrementuje wskaźnik

4. Jak prawidłowo zadeklarować wskaźnik na zmienną typu `int`?

A) `int p*;`
B) `int* p;`
C) `*int p;`
D) `pointer<int> p;`

5. Co wypisze poniższy kod?

```
int x = 10;
int* p = &x;
*p = 20;
cout << x;
```

A) 10
B) 20
C) Adres zmiennej x
D) Błąd kompilacji

6. Czym jest `nullptr`?

A) Wskaźnikiem na wartość zerową
B) Bezpieczną wartością inicjalizującą wskaźnik (pusty wskaźnik)
C) Funkcją zwracającą adres
D) Typem zmiennej

7. Czy nazwa tablicy w C++ jest wskaźnikiem na jej pierwszy element?

A) Tak, zawsze
B) Nie, nigdy
C) Tylko dla tablic dynamicznych
D) Tylko dla tablic znakowych

8. Co jest równoważne z `A[3]` w notacji wskaźnikowej?

A) `*A + 3`
B) `*(A + 3)`
C) `&A[3]`
D) `A + *3`

9. O ile bajtów przesuwa się wskaźnik `int*` po operacji `p++`?

A) 1 bajt
B) 2 bajty
C) 4 bajty (sizeof(int))
D) 8 bajtów

10. Jak przekazać zmienną do funkcji, aby funkcja mogła ją zmodyfikować?

A) Przez wartość: `void func(int x)`
B) Przez wskaźnik: `void func(int* x)`
C) Przez kopię: `void func(int& x)`
D) Nie można modyfikować zmiennych w funkcjach

11. Co robi funkcja `swap_int(int* a, int* b)` wywołana jako `swap_int(&x, &y)`?

A) Wypisuje wartości x i y
B) Zamienia wartości zmiennych x i y
C) Dodaje x do y
D) Nic, bo nie można modyfikować zmiennych przez wskaźniki

12. Który kod powoduje wyciek pamięci (memory leak)?

A) `int* p = new int(42); delete p;`
B) `int* p = new int[10]; delete[] p;`
C) `int* p = new int[10];` (bez delete)
D) `int x = 42; int* p = &x;`

13. Jak poprawnie zwolnić pamięć zaalokowaną dla tablicy?

A) `delete p;`
B) `delete[] p;`
C) `free(p);`
D) `remove p;`

14. Co wypisze poniższy kod?

```
int A[3] = {10, 20, 30};
cout << *(A + 1);
```

A) 10
B) 20
C) 30
D) Adres A[1]

15. Gdzie jest alokowana pamięć przy `int* p = new int(42);`?

A) Na stosie (stack)
B) Na stercie (heap)
C) W pamięci statycznej
D) W rejestrach procesora

16. Co się stanie przy próbie użycia niezainicjalizowanego wskaźnika?

```
int* p;
*p = 42;
```

A) Wartość 42 zostanie zapisana bezpiecznie
B) Program może ulec awarii (undefined behavior)
C) Kompilator zgłosi błąd
D) Wskaźnik automatycznie zostanie zainicjalizowany na nullptr

17. Jak iterować przez napis C-style wskaźnikiem?

A) `while (*p != 0)`
B) `while (*p != '\0')`
C) `while (p != NULL)`
D) `while (p < end)`

18. Co zwraca wyrażenie `p2 - p1` dla dwóch wskaźników tego samego typu?

A) Różnicę adresów w bajtach
B) Liczbę elementów między wskaźnikami
C) Sumę wskaźników
D) Błąd kompilacji

19. Który kod jest niebezpieczny?

A) `int* p = nullptr;`
B) `int x = 10; int* p = &x;`
C) `int* p = new int(42); delete p;`
D) `int* get_ptr() { int x = 10; return &x; }`

20. Jaka jest różnica między `int q = 42;` a `int* q = new int(42);`?

A) Brak różnicy, to samo
B) Pierwszy tworzy zmienną na stosie, drugi na stercie
C) Pierwszy jest szybszy, drugi wolniejszy
D) B i C są prawdziwe