

3.1 Assignment 1

Dòng cout << b << endl; -> in ra địa chỉ của biến a (**0x6efe1c**)

Dòng cout << *b << endl; -> in ra giá trị của biến a (**3**).

Dòng cout << &b << endl; -> in ra địa chỉ của biến b (**0x6efe10**)

Dòng cout << a << endl; -> in ra giá trị của biến a (**3**)

Dòng cout << &a << endl; -> in ra địa chỉ của biến a (**0x6efe1c**)

3.2 Assignment 2

```
int x,z;
```

```
float y;
```

```
char ch, *chp;
```

```
int *ip1, *ip2;
```

```
float *fp;
```

```
x = 100;
```

```
y = 20.0;
```

```
z = 50;
```

```
ch = 'Z';
```

```
ip1 = &x; // gán địa chỉ của biến x cho ip1
```

```
ip2 = &z; // gán địa chỉ của biến z cho ip2
```

```
fp = &y; // gán địa chỉ của biến y cho fp
```

```
chp = &ch; // gán địa chỉ của biến ch cho chp
```

```
ip2 = ip1; // gán giá trị của biến ip1 cho ip2
```

```
ip1 = &z; // gán địa chỉ của biến z cho ip1
```

```
*ip1 = *ip2; // gán giá trị của x cho z
```

```
*ip1 = 200; // gán giá trị của z thành 200
```

```
*ip1 = *ip2 + 300; // gán giá trị của z thành x + 300
```

```
*fp = 1.2; // gán giá trị của y thành 1.2
```

```
cout << x << endl; // in ra giá trị của biến x (100)
```

```
cout << y << endl; // in ra giá trị của biến y (1.2)
```

```

cout << z << endl; // in ra giá trị của biến z (400)
cout << ip1 << endl; // in ra địa chỉ của biến z (0x6efe18)
cout << *ip1 << endl; // in ra giá trị của biến z (400)
cout << &ip1 << endl; // in ra địa chỉ của biến ip1 (0x6efe00)
cout << ip2 << endl; // in ra địa chỉ của biến x (0x6efe1c)
cout << *ip2 << endl; // in ra giá trị của biến x (100)
cout << &ip2 << endl; // in ra địa chỉ của biến ip2 (0x6efd8)
cout << fp << endl; // in ra địa chỉ của biến y (0x6efe14)
cout << *fp << endl; // in ra giá trị của biến y (1.2)
cout << &fp << endl; // in ra địa chỉ của biến fp (0x6efd0)
cout << chp << endl; // in ra địa chỉ của biến ch (ZÜÖÖ?É☺)
cout << *chp << endl; // in ra giá trị của biến ch (Z)
cout << &chp << endl; // in ra địa chỉ của biến chp (0x6efe08)

```

3.3 Assignment 3

```

int *a = new int;
int *b = new int;
*a = 2;
b = a; // b và a trỏ vào cùng một ô địa chỉ
cout << *a << endl; // in ra 2
cout << *b << endl; // in ra 2
delete a;
delete b; // vì ô địa chỉ b trỏ vào cũng là địa chỉ a trỏ vào đã được xóa ở dòng trên nên sẽ xảy ra lỗi ở đây.

```

3.4 Assignment 4

```

int a = 3;
int *p = &a; // gán địa chỉ biến a cho p
cout << *p << endl; // in ra giá trị của biến a ( = 3 )
p = new int(5); // tạo một ô địa chỉ mới có giá trị = 5 và gán địa chỉ cho p
cout << *p << endl;

```

//Đoạn code này quên xóa ô địa chỉ đã tạo cho p nên sẽ xảy ra tràn bộ nhớ.

3.5 Assignment 5

1. `int v = 8, *r, *s; // v = 8`
2. `int *p;`
3. `int q = 100; // q = 100`
4. `p = &q; // *p = 100;`
5. `r = p; // *r = 100`
6. `*p = 20; // *p=20 → q = 20, *r = 20`
7. `p = new int;`
8. `*r = 30; //q = 30, *r = 30`
9. `q = v; // q = 8, *r = 8`
10. `s = p; // *s = *p`
11. `*s = 50; // *s = 50, *p = 50`

The last value of:

`*p = 50`

`q = 8`

`*r = 8`

`v = 8`

`*s = 50`

3.6 Assignment 6

1. `int *p, *q, v, nom[5];`
2. `p = &v;`
3. `*p = 12; // *p = 12, v = 12`
4. `q = p; // *q = 12`
5. `nom[0] = *q; // nom = [12,,,,]`
6. `p = nom; // *p = 12`
7. `p++; // *p = nom[1]`
8. `nom[2] = 12; // nom[2] = 12`

9. *p = 13; nom[1] = 13
10. *q = 10; /*q = 10, v = 10
11. v = 11; /*q = 11, v = 11
12. *(p+3) = 16; //nom[4] = 16
13. p = &nom[3];
14. *p = 10; // nom[3] = 10
15. p--; /*p = nom[2]

The last value of:

- *p = 12
- *q = 11
- v = 11
- nom[0] = 12
- nom[1] = 13
- nom[2] = 12
- nom[3] = 10
- nom[4] = 16

3.7 Assignment 7

```
#include<stdio.h>

int main()
{
    int *x;
    *x=100;
    return 0;
}
```

B. Error: suspicious pointer conversion at line “*x=100”.

3.9 Assignment 9

```
#include<stdio.h>

int main()
{
    char str[] = "peace";
    char *s = str;
    printf("%s\n", s++ +3);
    return 0;
}
```

D. ce

3.10 Assignment 10

```
#include<stdio.h>

int main()
{
    int i, a[] = {2, 4, 6, 8, 10};
    change(a, 5);
    for(i=0; i<=4; i++)
        printf("%d, ", a[i]);
    return 0;
}

void change(int *b, int n)
{
    int i;
    for(i=0; i<n; i++)
        *(b+1) = *(b+i)+5;
}
```

B. 2, 15, 6, 8, 10

3.11 Assignment 11

```
#include<stdio.h>

int main()
{
    int arr[] = {12, 13, 14, 15, 16};
    printf("%d, %d, %d\n", sizeof(arr), sizeof(*arr), sizeof(arr[0]));
    return 0;
}
```

B. 20, 4, 4

3.12 Assignment 12

```
#include<stdio.h>

int main()
{
    char *str;
    str = "%d\n";
    str++;
    str++;
    printf(str-2, 300);
    return 0;
}
```

D. 300

3.13 Assignment 13

2. The operator used for dereferencing or indirection is _____

- a) *
- b) &
- c) ->
- d) ->>

a) *

3.14 Assignment 14

3. Choose the right option

`string* x, y;`

- a) x is a pointer to a string, y is a string
- b) y is a pointer to a string, x is a string
- c) both x and y are pointers to string types
- d) none of the mentioned

a) x is a pointer to a string, y is a string

3.15 Assignment 15

4. Which one of the following is not a possible state for a pointer.

- a) hold the address of the specific object
- b) point one past the end of an object
- c) zero
- d) point to a type

d) point to a type

3.16 Assignment 16

5. Which of the following is illegal?

- a) `int *ip;`
- b) `string s, *sp = 0;`
- c) `int i; double* dp = &i;`
- d) `int *pi = 0;`

c) `int i; double* dp = &i;`

3.17 Assignment 17

6. What will happen in this code?

```
1.   int a = 100, b = 200;  
2.   int *p = &a, *q = &b;  
3.   p = q;
```

- a) b is assigned to a
- b) p now points to b
- c) a is assigned to b
- d) q now points to a

b) p now points to b

3.18 Assignment 18

7. What is the output of this program?

```
1.   #include <iostream>  
2.   using namespace std;  
3.   int main()  
4.   {  
5.       int a = 5, b = 10, c = 15;  
6.       int *arr[ ] = {&a, &b, &c};  
7.       cout << arr[1];  
8.       return 0;  
9.   }
```

- a) 5
- b) 10
- c) 15
- d) it will return some random number

d) it will return some random number

3.19 Assignment 19

9. What is the output of this program?

```
1.     #include <iostream>
2.     using namespace std;
3.     int main()
4.     {
5.         char arr[20];
6.         int i;
7.         for(i = 0; i < 10; i++)
8.             *(arr + i) = 65 + i;
9.         *(arr + i) = '\0';
10.        cout << arr;
11.        return(0);
12.    }
```

- a) ABCDEFGHIJ
- b) AAAAAAAAAA
- c) JJJJJJJJ
- d) None of the mentioned

a) ABCDEFGHIJ

3.20 Assignment 20

10. What is the output of this program?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      char *ptr;
6.      char Str[] = "abcdefg";
7.      ptr = Str;
8.      ptr += 5;
9.      cout << ptr;
10.     return 0;
11. }
```

- a) fg
- b) cdef
- c) defg
- d) abcd

a) fg

3.21 Assignment 21

MCQ: A pointer can be initialized with

- A. Null
- B. Zero
- C. Address of an object of same type
- D. All of them

D. All of them

3.22 Assignment 22

MCQ: Which from following is not a correct way to pass a pointer to a function?

- A. Non-constant pointer to non-constant data
- B. A non-constant pointer to constant data
- C. A constant pointer to non-constant data
- D. All of them

D. All of them

3.23 Assignment 23

MCQ: A qualifier that enables programmers to inform compiler that value of a particular variable should not be modified?

- A. ptr
- B. const
- C. stsr
- D. None of them

B. const

3.24 Assignment 24

C. The new operator

MCQ: Which operator returns address of unallocated blocks in memory?

- A. The delete operator
- B. The empty operator
- C. The new operator
- D. All of them

3.25 Assignment 25

MCQ: Referencing a value through a pointer is called

- A. Direct calling
- B. Indirection
- C. Pointer referencing
- D. All of them

B. Indirection

3.26 Assignment 26

MCQ: Which unary operator is used for determining size of an array?

- A. sizeof
- B. size_array
- C. s_array
- D. size_ofarray

A. sizeof

3.27 Assignment 27

MCQ: What is a Pointer?

- A. Pointer contains an address of a variable
- B. It's an operator
- C. It's a function
- D. None of them

A. Pointer contains an address of a variable

3.28 Assignment 28

MCQ: There are how many values that can be used to initialize a pointer

- A. 1
- B. 2
- C. 3
- D. 4

C. 3

3.29 Assignment 29


MCQ: A unary operator that returns address of its operands, are called

- A. Pointer operator
- B. Relationship operator
- C. Address operator
- D. Both A and B

C. Address operator

3.30 Assignment 30

1. What will be the output of the following program?



```
#include <iostream>
using namespace std;


int main()
{
    int a = 32, *ptr = &a;
    char ch = 'A', &cho = ch;

    cho += a;
    *ptr += ch;
    cout << a << ", " << ch << endl;
    return 0;
}
```

C. 129, a

3.31 Assignment 31

2. What will be the output of the following program?



```
#include <iostream>
using namespace std;

int main()
{
    const int i = 20;
    const int* const ptr = &i;
    (*ptr)++;
    int j = 15;
    ptr = &j;
    cout << i;
    return 0;
}
```

D. Compile error