BÀI TẬP TRẮC NGHIỆM LẬP TRÌNH C/C++

1 KHÁI NIỆM CƠ BẢN VỀ NGÔN NGỮ LẬP TRÌNH C/C++

Bài 1.1: Khi một chương trình kết thúc thành công thì sẽ trả về giá trị nào cho hệ điều hành?

a) 0

b) -1

c) 1

d) *không trả giá trị nào cả*

Bài 1.2: Một chương trình C phải có hàm nào?

a) `start()`

b) `system()`

c) `main()`

d) `program()`

Bài 1.3: Câu nào là cú pháp tạo khối lệnh?

a) `{ }`

b) `->` và `<-`

c) `BEGIN` và `END`

d) `(` và `)`

Bài 1.4: Câu nào là ký hiệu kết thúc một câu lệnh?

a) `.`

b) `;`

c) `:`

d) `'`

Bài 1.5: Câu nào là ghi chú?

a) `*/ Comments */`

b) `** Comment **`

c) `/* Comment */`

d) `{ Comment }`

Bài 1.6: Câu nào không phải là kiểu dữ liệu?

a) `float`

b) `real`

c) `int`

d) `double`

Bài 1.7: Câu nào là toán tử so sánh hai biến

a) `:=`

b) `=`

c) `equal`

d) `==`

Bài 1.8: Giá trị nào được xem là `true`

a) 1

b) 66

c) .1

d) -1

e) *tất cả câu trên*

Bài 1.9: Câu nào là toán tử luận lý "và"

a) `&`

b) `&&`

c) `|`

d) `|&`

Bài 1.10: Tính biểu thức `!(1 && !(0 || 1))`

a) `true`

b) `false`

c) *không tính được*

Bài 1.11: Câu nào là câu lệnh `if` đúng cú pháp

a) `if expression`

b) `if { expression`

c) `if ( expression )`

d) `expression if`

Bài 1.12: Giá trị cuối cùng của `x` sau khi thực thi xong đoạn chương trình?

```
int x;
for(x=0; x<10; x++)
{
}
```

a) 10

b) 9

c) 0

d) 1

Bài 1.13: Khi nào đoạn mã sau `while(x<100)` được thực thi?

a) Khi `x` nhỏ hơn một trăm

b) Khi `x` lớn hơn một trăm

c) Khi `x` bằng một trăm

Bài 1.14: Câu nào không phải cấu trúc lặp

a) `for`

b) `do while`

c) `while`

d) `repeat until`

Bài 1.15: Vòng lặp `while` thực hiện ít nhất bao nhiêu lần

a) 0

b) *vô tận*

c) 1

Bài 1.16: Câu nào không phải nguyên mẫu hàm?

a) `int funct(char x, char y);`

b) `double funct(char x)`

c) `void funct();`

d) `char x();`

Bài 1.17: Câu nào là kiểu trả về của nguyên mẫu hàm?

```
int func(char x, float v, double t);
```

a) `char`

b) `int`

c) `float`

d) `double`

Bài 1.18: Câu nào là một lời gọi hàm hợp lệ (giả sử hàm tồn tại)?

a) `funct;`

b) `funct x, y;`

c) `funct();`

d) `int funct();`

Bài 1.19: Câu nào là một hàm hoàn chỉnh?

a) `int funct();`

b) `int funct(int x){return x=x+1;}`

c) `void funct(int){printf( "Hello");`

d) `void funct(x){printf( "Hello"); }`

Bài 1.20: Câu nào theo sau câu lệnh `case`?

a) :

b) ;

c) -

d) *dòng mới*

Bài 1.21: Sau lệnh `case` thông thường là gì?

a) `end;`

b) `break;`

c) `stop;`

d) `;`

Bài 1.22: Từ khóa nào để xử lý những trường hợp còn lại?

a) `all`

b) `contingency`

c) `default`

d) `other`

Bài 1.23: Kết quả sau khi thực hiện đoạn chương trình sau?

```
int x=0;
switch(x)
{
    case 1: printf( "One" );
    case 0: printf( "Zero" );
    case 2: printf( "Hello World" );
}
```

a) One

b) Zero

c) Hello World

d) ZeroHello World

Bài 1.24: Câu nào là khai báo biến con trỏ

a) `int x;`

b) `int &x;`

c) `ptr x;`

d) `int *x;`

Bài 1.25: Câu nào trả về địa chỉ của biến `a`?

a) `*a;`

b) `a;`

c) `&a;`

d) `address(a);`

Bài 1.26: Câu nào trả về địa chỉ của biến do biến con trỏ `a` trỏ đến?

a) `a;`

b) `*a;`

c) `&a;`

d) `address(a);`

Bài 1.27: Câu nào trả về giá trị của biến do con trỏ `a` trỏ đến?

a) `a;`

b) `val(a);`

c) `*a;`

d) `&a;`

Bài 1.28: Câu nào là từ khóa hay hàm cấp phát bộ nhớ?

a) `new`

b) `malloc`

c) `create`

d) `value`

Bài 1.29: Câu nào là từ khóa hay hàm giải phóng bộ nhớ?

a) `free`

b) `delete`

c) `clear`

d) `remove`

Bài 1.30: Câu nào là truy xuất đến thành phần của biến cấu trúc `b`?

a) `b->var;`

b) `b.var;`

c) `b-var;`

d) `b>var;`

Bài 1.31: Câu nào truy xuất đến thành phần của biến cấu trúc do con trỏ `b` trỏ đến?

a) `b->var;`

b) `b.var;`

c) `b-var;`

d) `b>var;`

Bài 1.32: Câu định nghĩa một kiểu cấu trúc?

a) `struct {int a;}`

b) `struct a_struct {int a;}`

c) `struct a_struct int a;`

d) `struct a_struct {int a;};`

Bài 1.33: Câu nào khai báo biến cấu trúc kiểu `foo`?

a) `struct foo;`

b) `struct foo var;`

c) `foo;`

d) `int foo;`

Bài 1.34: Câu nào khai báo một biến mảng một chiều?

a) `int anarray[10];`

b) `int anarray;`

c) `anarray{10};`

d) `array anarray[10];`

Bài 1.35: Chỉ số của phần tử cuối cùng của biến mảng một chiều có 29 phần tử?

a) 29

b) 28

c) 0

d) *do lập trình viên quyết định*

Bài 1.36: Câu nào khai báo mảng hai chiều?

a) `array anarray[20][20];`

b) `int anarray[20][20];`

c) `int array[20, 20];`

d) `char array[20];`

Bài 1.37: Câu nào là truy xuất đến phần tử thứ bảy của mảng `foo` có 100 phần tử?

a) `foo[6];`

b) `foo[7];`

c) `foo(7);`

d) `foo;`

Bài 1.38: Câu nào trả về địa chỉ của phần tử đầu tiên của mảng `foo` có 100 phần tử?

a) `foo[0];`

b) `foo;`

c) `&foo;`

d) `foo[1];`

Bài 1.39: Câu nào là hằng chuỗi?

a) `Static String`

b) `"Static String"`

c) `'Static String'`

d) `char string[100];`

Bài 1.40: Kí tự nào kết thúc của một chuỗi?

a) `'.'`

b) `' '`

c) `'\0'`

d) `'/0'`

Bài 1.41: Câu nào đọc biến chuỗi 100 ký tự của biến chuỗi `x`?

a) `fgets(x, 101, stdin);`

b) `fgets(x, 100, stdin);`

c) `readline(x, 100, '\n');`

d) `read(x);`

Bài 1.42: Câu nào là so sánh hai chuỗi ký tự?

a) `compare(...);`

b) `stringcompare(...);`

c) `cmp(...);`

d) `strcmp(...);`

Bài 1.43: Câu nào thêm một chuỗi ký tự vào sau một chuỗi ký tự khác?

a) `append(...);`

b) `stringadd(...);`

c) `strcat(...);`

d) `stradd(...);`

2 BIẾN, TOÁN TỬ VÀ BIỂU THỨC

Bài 2.1: Kết quả của chương trình sau khi thực thi là? (biết kích thước kiểu `char` : 1 byte, `float` : 4 byte, `int` : 4 byte, `double` : 8 byte, `long` : 4 byte.)

```
#include <stdio.h>
int main()
{
    printf("%d\t",sizeof(6.5));
    printf("%d\t",sizeof(90000));
    printf("%d",sizeof('A'));
    return 0;
}
```

a) 8 4 1

b) 8 2 1

c) 4 4 1

d) *phụ thuộc vào trình biên dịch*

Bài 2.2: Kết quả chương trình sau khi thực thi?

```
#include <stdio.h>
int main()
{
    double num=5.2;
    int var=5;
    printf("%d\t",sizeof(!num));
    printf("%d\t",sizeof(var=15/2));
    printf("%d",var);
    return 0;
}
```

a) 1 4 5

b) 1 4 7

c) 8 4 7

d) *khác*

Bài 2.3: Kết quả đoạn chương trình sau khi thực thi

```
int w = 3;
int x = 31;
int y = 10;
double z = x / y % w;
printf("%f\n", z);
```

a) 1

b) 0

c) 0.1

Bài 2.4: Kết quả chương trình sau khi thực thi

```c
#include <stdio.h>
int main()
{
    char a=250;
    int expr;
    expr= a+ !a + ~a + ++a;
    printf("%d",expr);
    return 0;
}
```

a) -6

b) 4

c) 5

d) *khác*

Bài 2.5: Kết quả chương trình sau khi thực thi?

```c
#include <stdio.h>
int main()
{
    int a=-5;
    unsigned int b=-5u; // (*)
    if(a==b)
        printf("Avatar");
    else
        printf("Alien");
    return 0;
}
```

a) Avatar

b) Alien

c) *lỗi tại dòng* (*)

d) *khác*

Bài 2.6: Kết quả chương trình sau khi thực thi?

```c
#include <stdio.h>
#include <conio.h>
void main()
{
    int x = 3;
    printf("%d", x++ + ++x);
    getch();
}
```

a) 7

b) 8

c) 9

d) *khác*

Bài 2.7: Kết quả chương trình sau khi thực thi?

```c
void main()
{
    int i=5,j=6,k;
    k=i&j;
    printf("%d",k);
    getch();
}
```

a) 4

b) 0

c) 1

d) 5

Bài 2.8: Kết quả chương trình sau khi thực thi?

```
void main()
{
    int i=5,j=6;
    printf("%d", i | j);
    getch();
}
```

a) 7

b) 6

c) 5

d) 1

Bài 2.9: Kết quả chương trình sau khi thực thi?

```
#include <stdio.h>
#include "conio.h"
extern int x=0;
void main()
{
    x++;
    printf("%d",x);
    getch();
}
```

a) 0

b) 1

c) *lỗi* `x` *chưa được khai báo*

Bài 2.10: Kết quả đoạn chương trình sau khi thực thi?

```
extern int x=0;
void main()
{
    {
        int x=1;
    }
    printf("%d",x);
    getch();
}
```

a) 0

b) 1

c) *lỗi biên dịch*

Bài 2.11: Kết quả đoạn chương trình sau khi thực thi?

```
int y=0;
void main()
{
    {
        int x=0;
```

```
        x++;
        ++y;
    }
    printf("%d\t%d",x,y);
    getch();
}
```

a) 1 1

b) 1 0

c) *lỗi* `x` *chưa khai báo*

Bài 2.12: Kết quả đoạn chương trình sau khi thực thi?

```
void main()
{
    int x;
    {
        x++;
    }
    printf("%d",x);
    getch();
}
```

a) 1

b) 0

c) *lỗi biên dịch*

Bài 2.13: Kết quả đoạn chương trình sau khi thực thi?

```
void main()
{
    int x=0;
    {
        int x=0,y=0;
        y++;
        x++;
    }
    printf("%d",x);
    getch();
}
```

a) 1

b) *lỗi biên dịch*

c) 0

Bài 2.14: Kết quả đoạn chương trình sau khi thực thi?

```
void count()
{
    static int page=0;
    printf("%d",page);
    page++;
}
void main()
{
  int i;
  for(i=0;i<10;i++)
  {
    count();
  }
  getch();
}
```

a) 0123456789

b) 0000000000

c) 0101010101

Bài 2.15: Kết quả đoạn chương trình sau khi thực thi?

```
const int x = 5;
void main()
{
    int x[x];
    int y = sizeof(x) / sizeof(int);
    printf("%d",y);
    getch();
}
```

a) 1

b) 5

c) 20

d) *lỗi biên dịch*

Bài 2.16: Kết quả chương trình sau khi thực thi?

```
#include <stdio.h>
int main()
{
    int x=5,y=10,z=15;
    printf("%d %d %d");
    return 0;
}
```

a) *rác rác rác*

b) 5 10 15

c) 15 10 5

d) *lỗi chạy chương trình*

Bài 2.17: Kết quả chương trình sau khi thực thi?

```
#include <stdio.h>
int main()
{
    asm{
        mov bx,8;
        mov cx,10
        add bx,cx;
    }
    printf("%d",_BX);
    return 0;
}
```

a) 18

b) 8

c) 0

d) *lỗi biên dịch*

Bài 2.18: Kết quả đoạn chương trình sau khi thực thi?

```c
#include <stdio.h>
int main()
{
    char *url="c:\tc\bin\rw.c";
    printf("%s",url);
    return 0;
}
```

a) c:\tc\bin\rw.c

b) c:/tc/bin/rw.c

c) c: c inw.c

d) c:cinw.c

e) w.c in

Bài 2.19: Kết quả chương trình sau khi thực thi?

```c
#include <stdio.h>
int main()
{
    const int i=5;
    i++;
    printf("%d",i);
    return 0;
}
```

a) 5

b) 6

c) 0

d) *lỗi biên dịch*

Bài 2.20: Kết quả chương trình sau khi thực thi?

```c
#include <stdio.h>
#include <conio.h>
void main()
{
    char c=125;
    c=c+10;
    printf("%d",c);
    getch();
}
```

a) 135

b) 8

c) -121

d) 121

Bài 2.21: Kết quả chương trình sau khi thực thi?

```c
#include <stdio.h>
#include <conio.h>
int main()
{
    char c=48;
    int i, mask=01;
    for(i=1; i<=5; i++)
    {
```

```
        printf("%c", c|mask);
        mask = mask<<1;
    }
    getch();
}
```

a) 12480

b) 1248@

c) 12500

d) 12522

Bài 2.22: Kết quả đoạn chương trình sau khi thực thi?

```
#include <stdio.h>
#include <conio.h>
int main()
{
    float a = 0.7;
    if(0.7 > a)
        printf("Hi\n");
    else
        printf("Hello\n");
    getch();
}
```

a) Hi

b) Hello

c) *khác*

3 CÂU LỆNH ĐIỀU KIỆN

Bài 3.1: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int i = 3;
    if (!i) i++;
    i++;
    if (i==3) i+=2;
    i+=2;
    printf("%d\n", i);
    getch();
}
```

a) 7

b) 5

c) 6

d) *khác*

Bài 3.2: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int x;
    if(x=0)
        printf("Value of x is 0");
    else
        printf("Value of x is not 0");
    getch();
}
```

a) Value of x is 0

b) Value of x is not 0

c) *lỗi chương trình*

Bài 3.3: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int i;
    for(i=0;i<20;i++)
    {
        switch(i)
        {
            case 0:i+=5;
            case 1:i+=2;
            case 5:i+=5;
            default: i+=4; break;
        }
        printf("%d,",i);
    }
    getch();
}
```

a) 14,18,

b) 16,20,

c) 16,21,

Bài 3.4: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    static int i;
    while(i<=10&&i>=0)
        (i>2 ? i++:i--);
    printf("%d",i);
    getch();
}
```

a) -1

b) 0

c) *lỗi biên dịch chương trình*

Bài 3.5: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int i=10,j=20;
    if(i=20)
        printf(" Hello");
    else
        printf(" Hi");
    getch();
}
```

a) Hello

b) Hi

c) *lỗi biên dịch chương trình*

Bài 3.6: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int x=0,y=0;
    if(x==0||++y)
        printf(" x=%d",x);
    printf(" y=%d",y);
    getch();
}
```

a) x=0 y=1

b) x=0 y=0

c) *lỗi cú pháp*

Bài 3.7: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int i = 5, k;
    if (i == 0)
        goto label;
label:
    printf("%d", i);
```

```
    printf("Hey");
    getch();
}
```

a) Hey

b) 5

c) 5Hey

d) *lỗi biên dịch*

Bài 3.8: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    printf("%d ", 1);
    goto l1;
    printf("%d ", 2);
l1:
    goto l2;
    printf("%d ", 3);
l2:
    printf("%d ", 4);
    getch();
}
```

a) 1 4

b) 1 2 3 4

c) *lỗi cú pháp*

d) *khác*

Bài 3.9: Kết quả chương trình sau khi thực hiện?

```
#include <conio.h>
#include <stdio.h>
void foo();
int main()
{
    printf("%d ", 1);
    goto l1;
    printf("%d ", 2);
}
void foo()
{
l1:
    printf("3 ");
}
```

a) *lỗi biên dịch*

b) 3

c) 1

d) 1 3

Bài 3.10: Kết quả chương trình sau khi thực hiện?

```
#include <conio.h>
#include <stdio.h>
int main()
{
    int i = 0, j = 0;
```

```
    while (i < 2)
    {
l1:
        i++;
        while (j < 3)
        {
            printf("loop\n");
            goto l1;
        }
    }
    getch();
}
```

a) loop loop loop

b) *lặp vô tận*

c) *lỗi biên dịch*

Bài 3.11: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int i = 0, j = 0;
    while (l1: i < 2)
    {
        i++;
        while (j < 3)
        {
          printf("loop\n");
          goto l1;
        }
    }
    getch();
}
```

a) loop loop

b) loop loop loop loop

c) *lỗi biên dịch*

d) *khác*

Bài 3.12: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int a=15,b=10,c=5;
    if(a>b>c)
        printf("True");
    else
        printf("False");
    getch();
}
```

a) True

b) False

c) *lỗi biên dịch chương trình*

d) *lỗi chạy chương trình*

Bài 3.13: Kết quả chương trình sau khi thực hiện?

```c
#include <stdio.h>
#include <conio.h>
void main()
{
    int i = 0;
    switch (i)
    {
        case '0': printf("A"); break;
        case '1': printf("B"); break;
        default: printf("ABC");
    }
    getch();
}
```

a) A

b) B

c) ABC

Bài 3.14: Kết quả chương trình sau khi thực hiện?

```c
#include <stdio.h>
#include "conio.h"
void main()
{
    int i = 3;
    switch (i)
    {
        case 0+1: printf("A"); break;
        case 1+2: printf("B"); break;
        default: printf("ABC");
    }
    getch();
}
```

a) A

b) B

c) ABC

Bài 3.15: Kết quả chương trình sau khi thực hiện?

```c
#include <stdio.h>
#include <conio.h>
#define A 0
#define B 1
int main()
{
    int i = 3;
    switch (i & 1)
    {
        case A: printf("FALSE"); break;
        case B: printf("TRUE"); break;
        default: printf("Default");
    }
    getch();
}
```

a) FALSE

b) TRUE

c) Default

Bài 3.16: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int i;
    if (printf("0"))
        i = 3;
    else
        i = 5;
    printf("%d", i);
    getch();
}
```

a) 3

b) 5

c) 03

d) 05

Bài 3.17: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int a = 5;
    switch(a)
    {
        default: a = 4;
        case 6: a--;
        case 5: a = a+1;
        case 1: a = a-1;
    }
    printf("%d \n", a);
    getch();
}
```

a) 5

b) 4

c) 3

Bài 3.18: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int x = 3;
    if (x == 2);
    x = 0;
    if (x == 3)
        x++;
    else
        x += 2;
    printf("x = %d", x);
    getch();
}
```

a) x = 2

b) x = 6

c) x = 0

Bài 3.19: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int check = 20, arr[] = {10, 20, 30};
    switch (check)
    {
        case arr[0]: printf("A ");
        case arr[1]: printf("B");
        case arr[2]: printf("C");
    }
    getch();
}
```

a) ABC

b) BC

c) B

d) *lỗi biên dịch*

4 CÂU LỆNH LẶP

Bài 4.1: Hàm `func` trả về giá trị gì?

```
float func()
{
    int r = 0, d = 0, i=0;
    for (i; i < 2; i++)
    {
        r += 5 / d;
    }
    return r;
}
```

a) 5

b) 0

c) *lỗi*

d) *khác*

Bài 4.2: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    char s[]="man";
    int i;
    for(i=0;s[i];i++)
        printf("%c%c%c%c\t",s[i],*(s+i),*(i+s),i[s]);
    getch();
}
```

a) mmmm aaaa nnnn

b) mmm aaa nnn

c) mmmm aaa nnn

d) *khác*

Bài 4.3: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int i = 0;
    char ch = 'A';
    do {
        putchar (ch);
    } while(i++ < 5 || ++ch <= 'F');
    getch();
}
```

a) AAAAAABCDEF

b) AAAAAABCDE

c) ABCDEF

d) *khác*

Bài 4.4: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    int array[2][2] = {0, 1, 2, 3};
    int i;
    int sum = 0;
    for (i =0; i < 4; ++i)
    {
        int x, y;
        x = i % 2;
        if (x)
        {
            y = 0;
        }
        else
        {
            y = 1;
        }
        sum += array[x][y];
    }
    printf("%d\n", sum);
    getch();
}
```

a) 4

b) 5

c) 6

d) 3

Bài 4.5: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int k;
    for (k = -3; k < -5; k++)
        printf("Hello");
    getch();
}
```

a) Hello

b) Nothing

c) *lỗi biên dịch chương trình*

d) *lỗi chạy chương trình*

Bài 4.6: Kết quả chương trình sau khi thực hiện?

```
void main()
{
    double k = 0;
    for (k = 0.0; k < 3.0; k++);
        printf("%lf", k);
    getch();
}
```

a) 012

b) *lỗi chạy chương trình*

c) 3

d) 2

Bài 4.7: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
int main()
{
    int i = 0;
    for (; ; ;)
        printf("In for loop\n");
    printf("After loop\n");
}
```

a) *lỗi biên dịch chương trình*

b) Infinite Loop

c) Nothing

Bài 4.8: Kết quả chương trình sau khi thực hiện?

```
#include <conio.h>
#include <stdio.h>
int foo();
void main()
{
    int i = 0;
    for (foo(); i == 1; i = 2)
        printf("In for loop\n");
    printf("After loop\n");
    getch();
}
int foo()
{
  return 1;
}
```

a) In for loop

b) After loop

c) *lỗi biên dịch chương trình*

Bài 4.9: Kết quả chương trình sau khi thực hiện?

```
#include <conio.h>
#include <stdio.h>
int main()
{
    int i = 0;
    while (i = 0)
        printf("True\n");
    printf("False\n");
    getch();
}
```

a) True

b) False

c) *lỗi biên dịch chương trình*

d) *khác*

Bài 4.10: Kết quả chương trình sau khi thực hiện?

```c
#include <conio.h>
#include <stdio.h>
int main()
{
    int i = 0, j = 0;
    while (i < 5, j < 10)
    {
        i++;
        j++;
    }
    printf("%d, %d\n", i, j);
    getch();
}
```

a) 5, 5

b) 10, 10

c) *lỗi cú pháp*

Bài 4.11: Kết quả chương trình sau khi thực hiện?

```c
#include <conio.h>
#include <stdio.h>
int main()
{
    int a = 0, i = 0, b = 0 ;
    for (i = 0;i < 5; i++)
    {
        a++;
        continue;
        b++;
    }
    printf("\n a = %d,b =%d",a,b);
    getch();
}
```

a) a = 5,b = 5

b) a = 4,b = 4

c) a = 5,b = 0

d) *khác*

Bài 4.12: Kết quả chương trình sau khi thực hiện?

```c
void main()
{
  int i = 0;
  for (i = 0;i < 5; i++)
    if (i < 4)
    {
      printf("Hello");
      break;
    }
  getch();
}
```

a) Hello

b) Hello *được in 3 lần*

c) Hello *được in 4 lần*

d) Hello *được in 5 lần*

Bài 4.13: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i=0;
    for(;i<=2;)
        printf(" %d",++i);
    getch();
}
```

a) 1 2 3

b) 0 1 2 3

c) 0 1 2

Bài 4.14: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int x;
    for(x=1;x<=5;x++);
        printf("%d",x);
    getch();
}
```

a) 12345

b) 123456

c) 6

d) 1234

Bài 4.15: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int i = 3;
    while (i--)
    {
        int i = 100;
        i--;
        printf("%d ", i);
    }
    getch();
}
```

a) 99 99 99

b) *lỗi biên dịch chương trình*

c) 1

Bài 4.16: Chuỗi ”vncoding” được in bao nhiều lần?

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int i = 1024;
```

```
    for (; i; i >>= 1)
        printf("\nvncoding");
    getch();
}
```

a) 10

b) 11

c) *vô hạn*

Bài 4.17: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i=2,j=2;
    while(i+1?--i:j++)
        printf("%d",i);
    getch();
}
```

a) 1

b) 2

c) lỗi biên dịch chương trình

Bài 4.18: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i,j;
    i=j=2;
    while(--i&&j++)
        printf("%d %d",i,j);
    getch();
}
```

a) 1 3

b) 1 2

c) *không có gì*

Bài 4.19: Kết quả chương trình sau khi thực hiện?

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int x=011,i;
    for(i=0;i<x;i+=3)
    {
        printf("Start ");
        continue;
        printf("End");
    }
    getch();
}
```

a) Start End Start End

b) Start Start Start

c) Start Start Start Start

5 MẢNG, CHUỖI VÀ CON TRỎ

Bài 5.1: Kết quả đoạn chương trình sau khi thực hiện?

```
void myfunc(char** param)
{

}
void main()
{
    char* string = (char*)malloc(64);
    strcpy(string, "hello_World");
    myfunc(&string);
    myfunc(&string);
    printf("%s\n", string);
    getch();
}
```

a) hello_World

b) ello_World

c) llo_World

d) lo_World

Bài 5.2: Kết quả đoạn chương trình sau khi thực hiện?

```
void myfunc(char** param)
{
    ++*param;
}
void main()
{
    char* string = (char*)malloc(64);
    strcpy(string, "hello_World");
    myfunc(&string);
    myfunc(&string);
    printf("%s\n", string);
    getch();
}
```

a) hello_World

b) ello_World

c) llo_World

d) lo_World

Bài 5.3: Kết quả đoạn chương trình sau khi thực hiện?

```
void main()
{
    int ints[] = { 0, 1, 2, 3 };
    int* i1 = ints + 1;
    int a = ++*i1;
    int b = a + *i1;
    printf("%d\n", b);
    getch();
}
```

a) 4

b) 3

c) 6

Bài 5.4: Kết quả đoạn chương trình sau khi thực hiện?

```
int main()
{
    int ints[] = { 0, 5, 10, 15 };
    int* i2 = ints + 2;
    int a = *i2++; // <=> a = *(i2++);
    printf("%d#%d\n", a, *i2);
    getch();
}
```

a) 10#15

b) 10#10

c) 15#15

d) 11#15

Bài 5.5: Kết quả đoạn chương trình sau khi thực hiện?

```
int main()
{
    int ints[] = { 0, 5, 10, 15 };
    int* i2 = ints + 2;
    int a = *++i2;
    printf("%d#%d\n", a, *i2);
    getch();
}
```

a) 10#15

b) 10#10

c) 15#15

d) 11#15

Bài 5.6: Kết quả đoạn chương trình sau khi thực hiện?

```
void main()
{
    int ints[] = { 0, 1, 2, 3 };
    int* i1 = ints + 1;
    int* i2 = ints + 2;
    int a = ++*i1 + *i2++;
    int b = *++i1 + *i2--;
    printf("%d#%d", a, b);
    getch();
}
```

a) 4#4

b) 4#5

c) 5#6

d) 4#6

Bài 5.7: Kết quả đoạn chương trình sau khi thực hiện?

```c
void main()
{
    int i=400;
    int *ptr=&i;
    *++ptr=2;
    printf("%d %d",i,*ptr);
    getch();
}
```

a) 400 2

b) 400 400

c) 400 401

d) *lỗi biên dịch*

Bài 5.8: Kết quả đoạn chương trình sau khi thực hiện?

```c
void main()
{
    char str[]={"pvpit"};
    char *s1=str;
    s1++;
    printf("%c",*s1);
    getch();
}
```

a) pvpit

b) vpit

c) v

d) *khác*

Bài 5.9: Kết quả đoạn chương trình sau khi thực hiện?

```c
void main()
{
    char *s="\12345s\n";
    printf("%d",strlen(s));
    printf("\n%s",s);
    getch();
}
```

a) 5

b) 7

c) 9

d) 10

Bài 5.10: Dòng nào trong đoạn chương trình sau bị báo lỗi bởi trình biên dịch

```c
1 int main(int argc, char** argv)
2 {
3     const char* foo = "wow";
4     foo = "top";
5     foo[0] = 1;
6     return 0;
7 }
```

a) 2

b) 3

c) 4

d) *không có dòng nào*

Bài 5.11: Kết quả đoạn chương trình sau khi thực hiện?

```
void main()
{
    int x = 5,y = 6;
    int* const p=&x;
    p = &y;
    printf("%d",(*p));
    getch();
}
```

a) *lỗi biên dịch*

b) 6

c) 5

d) *khác*

Bài 5.12: Kết quả đoạn chương trình sau khi thực hiện?

```
void main()
{
    int x = 5,y = 8;
    const int* p;
    p = &x;
    p = &y;
    x++;
    printf("%d",*p);
    getch();
}
```

a) 5

b) 6

c) 8

d) *lỗi biên dịch*

Bài 5.13: Kết quả đoạn chương trình sau khi thực hiện?

```
void main()
{
    int x = 5;
    const int* p;
    p = &x;
    x++;
    *p = 4;
    printf("%d",*p);
    getch();
}
```

a) 5

b) 6

c) 4

d) *lỗi biên dịch*

Bài 5.14: Kết quả đoạn chương trình sau khi thực hiện?

```c
#include<stdio.h>
int main()
{
    int a = 320;
    char *ptr;
    ptr =( char *)&a;
    printf("%d ",*ptr);
    return 0;
}
```

a) 320

b) 64

c) *lỗi biên dịch*

Bài 5.15: Kết quả đoạn chương trình sau khi thực hiện?

```c
#include<stdio.h>
int main()
{
    int i = 3;
    int *j;
    int **k;
    j=&i;
    k=&j;
    printf("%u , %u , %d ",k,*k,**k);
    return 0;
}
```

a) *địa chỉ của* `j` , *địa chỉ của* `i` , 3

b) *lỗi biên dịch*

c) 3 , 3 , 3

Bài 5.16: Kết quả đoạn chương trình sau khi thực hiện?

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
    char *ptr1 = NULL;
    char *ptr2 = 0;
    printf("\n%d",ptr2);
    strcpy(ptr1," c");
    strcpy(ptr2,"questions");
    printf("\n%s %s",ptr1,ptr2);
    getch();
}
```

Bài 5.17: Kết quả đoạn chương trình sau khi thực hiện?

```c
#include<stdio.h>
#include<conio.h>
int main()
{
    int a = 10;
    void *p = &a;
    int *ptr = p;
    printf("%u\n",*ptr);
    getch();
}
```

Bài 5.18: Kết quả đoạn chương trình sau khi thực hiện?

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a = 5,b = 10,c;
    int *p = &a,*q = &b;
    c = p - q;
    printf("%d" , c);
    getch();
}
```

Bài 5.19: Kết quả đoạn chương trình sau khi thực hiện?

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i = 5 , j;
    int *p , *q;
    p = &i;
    q = &j;
    j = 5;
    printf("%d %d",*p,*q);
    getch();
}
```

a) 5 5

b) *lỗi biên dịch*

c) 5 *rác*

Bài 5.20: Kết quả đoạn chương trình sau khi thực hiện?

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i = 5;
    int *p;
    p = &i;
    printf(" %u %u", *&p , &*p);
    getch();
}
```

a) *địa chỉ của* `i` *địa chỉ của* `i`

b) *rác*

c) *lỗi biên dịch*

Bài 5.21: Kết quả đoạn chương trình sau khi thực hiện?

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int array[2][2][3]={0,1,2,3,4,5,6,7,8,9,10,11};
    printf("%d",array[1][0][2]);
    getch();
}
```

a) 6

b) 7

c) 8

d) 9

Bài 5.22: Kết quả đoạn chương trình sau khi thực hiện?

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char arr[8]={'V','I','E','T','N','A','M'};
    char *p;
    p=(char *)(arr+2)[2];
    printf("%c",p);
    getch();
}
```

a) I

b) E

c) M

d) N

Bài 5.23: Kết quả đoạn chương trình sau khi thực hiện?

```
#include "stdio.h"
#include "conio.h"
void main()
{
    char ch[]={'0','1','2','3','4','5','6','7','8','9'};
    int *p=(int*)ch;
    p++;
    printf("%x",*p);
    getch();
}
```

a) 37363534

b) 34353637

c) 45673333

Bài 5.24: Kết quả đoạn chương trình sau khi thực hiện?

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i=11;
    int const * p=&i;
    p++;
    printf("%d",*p);
    getch();
}
```

a) 11

b) 12

c) *rác*

d) *lỗi biên dịch*

Bài 5.25: Phát biểu nào sau là phát biểu đúng

a) The array int num[26]; can store 26 elements

b) The expression num[1] designates the very first element in the array

c) It is necessary to initialize the array at the time of declaration.

d) The declaration num[SIZE] is allowed if SIZE is a macro.

a) 1,4

b) 3

c) 1,2

d) 1

Bài 5.26: Hàm thư viện tìm kiếm ký tự xuất hiện cuối cùng trong chuỗi

a) strnstr()

b) strrchr()

c) laststr()

d) strstr()

Bài 5.27: Kết quả đoạn chương trình sau khi thực hiện? (giả sử địa chỉ của mảng `a` bắt đầu tại 1002 và kích thước của `int` là 4 bytes)

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a[3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
    printf("%u, %u, %u\n", a[0]+1, *(a[0]+1), *(*(a+0)+1));
    getch();
}
```

a) 1006, 2, 2

b) 1006, 4, 4

c) 1002, 5, 5

d) *lỗi*

Bài 5.28: Khai báo sau nghĩa là gì?

```
int (*ptr)[10];
```

a) `ptr` là mảng con trỏ đến 10 số nguyên is array of pointers to 10 integers

b) `ptr` là con trỏ đến mảng có 10 phần tử số nguyên is a pointer to an array of 10 integers

c) `ptr` là mảng mảng có 10 phần tử số nguyên

d) `ptr` là một con trỏ đến mảng

Bài 5.29: Kết quả đoạn chương trình sau khi thực hiện?

```
#include<stdio.h>
#include<conio.h>
int main()
{
    char str[] = "VNCODING\0\.NET\0";
    printf("%s\n", str);
    getch();
}
```

a) VNCODING

b) VNCODING\0\.NET\0

c) VNCODING\0\.NET

Bài 5.30: Kết quả đoạn chương trình sau khi thực hiện?

```
#include<stdio.h>
#include<conio.h>
void swap(char *, char *);
int main()
{
    char *pstr[2] = {"VNCODING", ".NET"};
    swap(pstr[0], pstr[1]);
    printf("%s%s", pstr[0], pstr[1]);
    getch();
}
void swap(char *t1, char *t2)
{
    char *t;
    t=t1;
    t1=t2;
    t2=t;
}
```

a) VNCODING.NET

b) .NETVNCODING

c) *địa chỉ của* `pstr[0]` *địa chỉ của* `pstr[1]`

Bài 5.31: Kết quả đoạn chương trình sau khi thực hiện?

```
#include<stdio.h>
#include<conio.h>
void swap(char **, char **);
int main()
{
    char *pstr[2] = {"VNCODING", ".NET"};
    swap(&pstr[0], &pstr[1]);
    printf("%s%s", pstr[0], pstr[1]);
    getch();
}
void swap(char **t1, char **t2)
{
    char *t;
    t=*t1;
    *t1=*t2;
    *t2=t;
}
```

a) VNCODING.NET

b) .NETVNCODING

c) *địa chỉ của* `pstr[0]` *địa chỉ của* `pstr[1]`

6 KIỂU CẤU TRÚC

Bài 6.1: Kết quả chương trình sau khi thực thi?

```
#include "stdio.h"
#include "conio.h"
typedef struct {
    char c; // 1 byte
    float b; // 4 byte
    int a; // 4 byte
} A;
void main()
{
    printf("\n Size of struct: %d",sizeof(A));
    getch();
}
```

a) 9

b) 12

c) 16

d) 24

Bài 6.2: Kết quả chương trình sau khi thực thi?

```
#include "stdio.h"
#include "conio.h"
typedef struct {
    int a[2]; // 8 byte
    char b[5]; // 5 byte
    char c[5]; // 5 byte
} A;
void main()
{
    printf("\n Size of struct: %d",sizeof(A));
    getch();
}
```

a) 20

b) 18

c) 32

d) 24

Bài 6.3: Kết quả chương trình sau khi thực thi?

```
#include "stdio.h"
#include "conio.h"
struct birthday {
    int d; // day
    int m; // month
    int y; // year
};
struct info {
    int ID; // code of staff
    birthday b;
```

```
};
void main()
{
    info a = {1009,16,9,1989};
    printf("\nID=%d, dd/mm/yyyy = %d/%d/%d",a.ID,a.b.d,a.b.m,a.b.y);
    getch();
}
```

a) ID=1009, dd/mm/yyyy = 16/09/1989

b) ID = 1009, dd/mm/yyyy = *rác/rác/rác*

c) *lỗi cú pháp*

7 MACRO

Bài 7.1: Kết quả chương trình sau khi thực thi?

```
#include<stdio.h>
#include<conio.h>
#define x 5+2
void main()
{
    int i;
    i=x*x*x;
    printf("%d",i);
    getch();
}
```

a) 21

b) 27

c) *lỗi biên dịch*

d) *khác*

Bài 7.2: Kết quả chương trình sau khi thực thi?

```
#include<stdio.h>
#include<conio.h>
#define call(x,y) x##y
void main()
{
    int x=5,y=10,xy=20;
    printf("%d",xy+call(x,y));
    getch();
}
```

a) 530

b) 70

c) 40

d) *lỗi biên dịch*

8 KIỂU DỮ LIỆU DANH SÁCH LIÊN KẾT

Bài 8.1: What does the following function do for a given Linked List with first node as head?

```
void fun1(struct node* head)
{
    if(head == NULL) return;
    fun1(head->next);
    printf("%d ", head->data);
}
```

a) Prints all nodes of linked lists

b) Prints all nodes of linked list in reverse order

c) Prints alternate nodes of Linked List

d) Prints alternate nodes in reverse order

Bài 8.2: Which of the following points is/are true about Linked List data structure when it is compared with array

a) Arrays have better cache locality that can make them better in terms of performance.

b) It is easy to insert and delete elements in Linked List

c) Random access is not allowed in a typical implementation of Linked Lists

d) The size of array has to be pre-decided, linked lists can change their size any time.

e) All of the above

Bài 8.3: Consider the following function that takes reference to head of a Doubly Linked List as parameter. Assume that a node of doubly linked list has previous pointer as prev and next pointer as next.

```
void fun(struct node **head_ref)
{
    struct node *temp = NULL;
    struct node *current = *head_ref;
    while (current != NULL)
    {
        temp = current->prev;
        current->prev = current->next;
        current->next = temp;
        current = current->prev;
    }
    if(temp != NULL )
        *head_ref = temp->prev;
}
```

Assume that reference of head of following doubly linked list is passed to above function `1 <--> 2 <--> 3 <--> 4 <--> 5 <--> 6`. What should be the modified linked list after the function call?

a) `2 <--> 1 <--> 4 <--> 3 <--> 6 <--> 5`

b) `5 <--> 4 <--> 3 <--> 2 <--> 1 <--> 6`

c) `6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1`

d) `6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2`

Bài 8.4: Which of the following sorting algorithms can be used to sort a random linked list with minimum time complexity?

a) Insertion Sort

b) Quick Sort

c) Heap Sort

d) Merge Sort

Bài 8.5: The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

```
/* Link list node */
struct node {
    int data;
    struct node* next;
};
/* head_ref is a double pointer which points to head (or start) pointer
   of linked list */
static void reverse(struct node** head_ref)
{
    struct node* prev = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    /*ADD A STATEMENT HERE*/
}
```

What should be added in place of `/*ADD A STATEMENT HERE*/`, so that the function correctly reverses a linked list.

a) `*head_ref = prev;`

b) `*head_ref = current;`

c) `*head_ref = next;`

d) `*head_ref = NULL;`

Bài 8.6: What is the output of following function for start pointing to first node of following linked list? `1->2->3->4->5->6`

```
void fun(struct node* start)
{
    if(start == NULL)
        return;
    printf("%d ", start->data);
    if(start->next != NULL )
    fun(start->next->next);
    printf("%d ", start->data);
}
```

a) `1 4 6 6 4 1`

b) `1 3 5 1 3 5`

c) `1 2 3 5`

d) `1 3 5 5 3 1`

Bài 8.7: The following C function takes a simply-linked list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.

```
typedef struct node
{
    int value;
    struct node *next;
} Node;
Node *move_to_front(Node *head)
{
    Node *p, *q;
    if ((head == NULL: || (head->next == NULL))
        return head;
    q = NULL;
    p = head;
    while (p-> next !=NULL)
    {
        q = p;
        p = p->next;
    }
    // missing code here
    return head;
}
```

a) `q = NULL; p->next = head; head = p;`

b) `q->next = NULL; head = p; p->next = head;`

c) `head = p; p->next = q; q->next = NULL;`

d) `q->next = NULL; p->next = head; head = p;`

Bài 8.8: The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node
{
    int value;
    struct node *next;
};
void rearrange(struct node *list)
{
    struct node *p, * q;
    int temp;
    if ((!list) || !list->next)
        return;
    p = list;
    q = list->next;
    while(q)
    {
        temp = p->value;
        p->value = q->value;
        q->value = temp;
```

```
        p = q->next;
        q = p?p->next:0;
    }
}
```

a) 1,2,3,4,5,6,7

b) 2,1,4,3,6,5,7

c) 1,3,2,5,4,7,6

d) 2,3,4,5,6,7,1

Bài 8.9: In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is

a) $log2n$

b) $n/2$

c) $log2n$–1

d) n

Bài 8.10: Suppose each set is represented as a linked list with elements in arbitrary order. Which of the operations among union, intersection, membership, cardinality will be the slowest?

a) union only

b) intersection, membership

c) membership, cardinality

d) union, intersection

Bài 8.11: Consider the function `f` defined below.

```
struct item
{
    int data;
    struct item * next;
};
int f(struct item *p)
{
    return ( (p == NULL) || (p->next == NULL) ||
             (( p->data <= p->next->data) && f(p->next)) );
}
```

For a given linked list p, the function f returns 1 if and only if

a) the list is empty or has exactly one element

b) the elements in the list are sorted in non-decreasing order of data value

c) the elements in the list are sorted in non-increasing order of data value

d) not all elements in the list have the same data value.

Bài 8.12: A circularly linked list is used to represent a Queue. A single variable p is used to access the Queue. To which node should p point such that both the operations `enQueue` and `deQueue` can be performed in constant time?

a) rear node

b) front node

c) not possible with a single pointer

d) node next to front

Bài 8.13: What are the time complexities of finding 8th element from beginning and 8th element from end in a singly linked list? Let n be the number of nodes in linked list, you may assume that $n > 8$.

a) $O(1)$ and $O(n)$

b) $O(1)$ and $O(1)$

c) $O(n)$ and $O(1)$

d) $O(n)$ and $O(n)$

Bài 8.14: Is it possible to create a doubly linked list using only one pointer with every node.

a) Not Possible

b) Yes, possible by storing XOR of addresses of previous and next nodes.

c) Yes, possible by storing XOR of current node and next node

d) Yes, possible by storing XOR of current node and previous node

Bài 8.15: Given pointer to a node X in a singly linked list. Only one pointer is given, pointer to head node is not given, can we delete the node X from given linked list?

a) Possible if X is not last node. Use following two steps (a) Copy the data of next of X to X. (b) Delete next of X.

b) Possible if size of linked list is even.

c) Possible if size of linked list is odd

d) Possible if X is not first node. Use following two steps (a) Copy the data of next of X to X. (b) Delete next of X.

Bài 8.16: You are given pointers to first and last nodes of a singly linked list, which of the following operations are dependent on the length of the linked list?

a) Delete the first element

b) Insert a new element as a first element

c) Delete the last element of the list

d) Add a new element at the end of the list

Bài 8.17: Consider the following function to traverse a linked list.

```
void traverse(struct Node *head)
{
    while (head->next != NULL)
    {
        printf("%d ", head->data);
        head = head->next;
    }
}
```

Which of the following is FALSE about above function?

a) The function may crash when the linked list is empty

b) The function doesn't print the last node when the linked list is not empty

c) The function is implemented incorrectly because it changes head

Bài 8.18: In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is

a) $log2n$

b) $n/2$

c) $log2n - 1$

d) n