Internals - 01

OOC - Object oriented concepts

Shahulhameed. S
1FN18CS097
CSE A'Sec

1a.

POP	OOP
• Emphasis on procedure.	• Emphasis on data.
• programming task is divided into collecting data structures and functions.	• programming task is divided into tasks.
• procedures are being exported from data being manipulated	• procedures are not exported from data, instead procedure and data are combined together.
• Data is not ~~secure~~ secure	• Data is secure
• A piece of code uses the data to perform the specific task	• The data uses the piece of code to perform specific task.
• Top down approach is used in the program design.	• Bottom down approach is used in program design.
• Debugging is the difficult ~~as~~ the Code size increases.	• ~~Debugging~~ Debugging is ~~is~~ Easy Even the code size increases.

2. The controll will move from calling to called function. Then arguments will be pushed on to the stack. then Controll will move back to the calling from called function. This process takes extra time in execution.

To avoid this, we use inline function.

when the function is declared as inline, compiler replaces a function call with function code.

Ex:
```
#include <iostream.h>
void main()
{
  cout<< max (40,60);
  getch();
}
inline int max(int a, int b)
  {
    if (a > b)
        return a;
    else
        return b;
  }
```

3a. A reference variable is nothing but a reference for an existing variable. It shares the memory location with an existing variable.

Syntax:-

<data_type> &<ref_var_name> = <existing_var_name>;

If a is an existing integer type and we want to declare ref as a reference to it, the statement is as follows

int & iref = a;

iref is a reference to 'a', this means that although iref and 'a' have separate entries in the O.S, their addresses are actually the same. Thus, a change in the value of 'a' will naturally reflect in iref and vice versa

program:

```
#include <iostream.h>
using namespace std;
int main()
{
 int a, b, temp;
 cout<<" Enter a number :\n" <<endl;
 cin >> a;
 cout<<" Enter another number :\n" << endl;
 cin >>b;
 temp =a;
 a = b;
 b =temp;
 cout<<" Numbers before swapping :\n" <<endl;
 cout<<" a = " <<a << " b = " << b<<endl;
 cout<<" Numbers after Swapping :\n" <<endl;
 cout<<" a = " << a << " b = " << b <<endl;
 return 0;
 }
```

(3b) Two or more functions having same name but different argument list, the arguments may differ in the type or number, or both.

However, the return types of overloaded methods can be the same or different is called function overloading.

```
#include <iostream.h>
#include <stdlib.h>
#define pi 3.14.
class Fno
{
  public:
    void area (int);
    void area(int, int);
    void area (float, int, int);
};
void Fno :: area(int &)
{
  cout<< " Area of circle:\n"<< pi * & * &;
}
void Fno :: area(int a, int b)
{
  cout <<" area of rectangle: \n" << a*b;
}
void Fno :: area (float *, int a, int b)
{
  cout<< " Area of triangle :\n"<< * * a* b;
}
void main()
{
  int ch;
  int a, b, r;
  clrscr();
```

```
Fno obj';
cout<<"\n1. Area of circle \n2. Area of rectangle \n3.
        Area of triangle \n4. Exit :";
cout<<"Enter your choice :";
cin>>ch;
switch(ch)
{
case 1: cout<<" Enter radius of the circle:";
  cin>>r;
      obj.area(r);
      break;
case 2: cout<<" Enter sides of rectangle:";
      cin>>a>>b;
       obj.area(a,b);
       break;
case 3: cout<<" Enter sides of the triangle!";
        cin>>a>>b;
         obj.area(0.5,a,b);
        break;
case 4: Exit(0);
default: cout<<" Invalid choice!!!\n";
 }
getch();
}
```