Data Structure > How you are storing your data

Arrays

Collections

Stack

Queue

Linked List

Trees

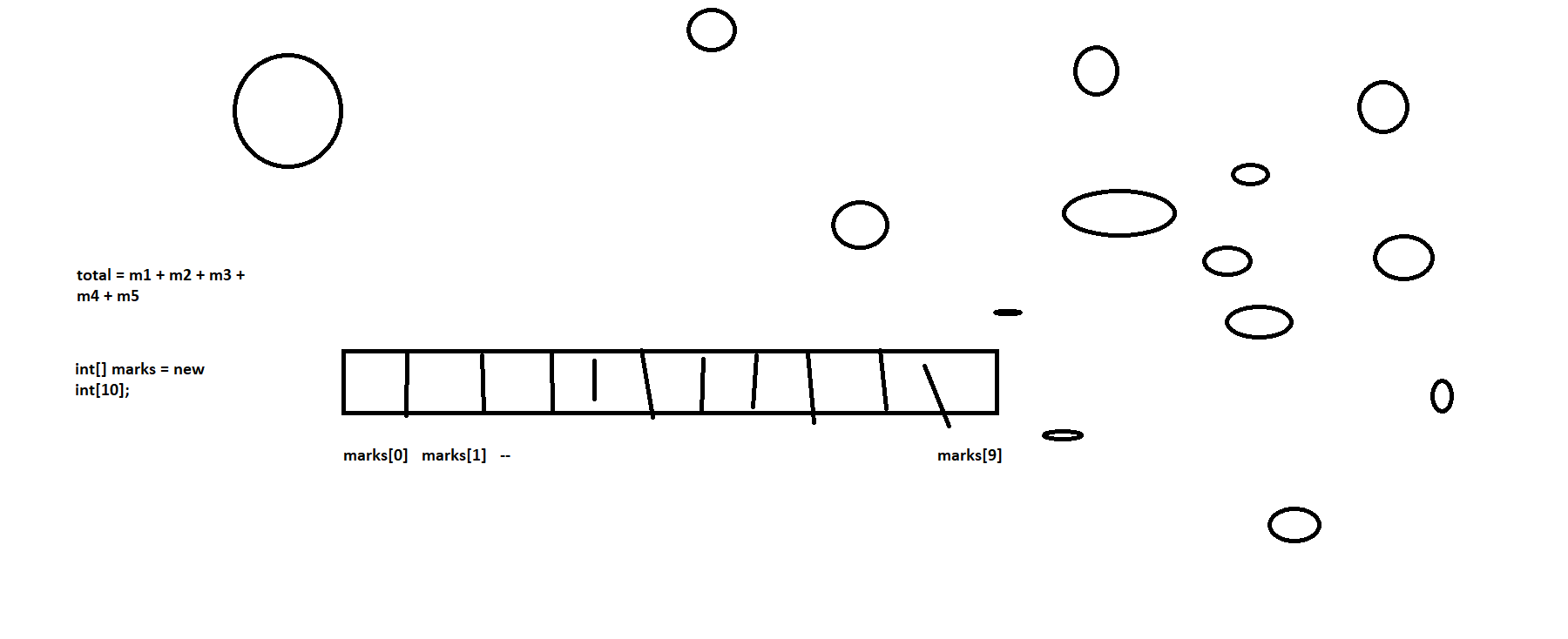
Graphs

Heap

Matrix

Arrays

Int marks1, marks2, ………………………………….. marks10



Advantages of Arrays

1. Single name
2. No need to declare different variables
3. All the elements are in continuation
4. All the elements are of same type

Disadvantages of Arrays

1. Insertion / deletion will be time consuming, because it requires lots of reshuffling of elements
2. No deletion , we cannot delete elements
3. Static Allocation of memory, before we use array , we should know its size
4. Memory could be wasted, when we do not insert elements

using System;

namespace LinearDataStructureDemos

{

class Program

{

static int[] number = new int[] { 10, 20, 30, 40, 50, 60, 0, 0, 0, 0, 0 };

static void Main(string[] args)

{

string choice = "y";

while (choice == "y")

{

int ch = MainMenu();

switch (ch)

{

case 1:

{

Console.WriteLine("Enter Element to insert");

int num = int.Parse(Console.ReadLine());

Insert(num);

break;

}

case 2:

{

Console.WriteLine("Enter Element to delete");

int num = int.Parse(Console.ReadLine());

Delete(num);

break;

}

case 3:

{

Display(number);

break;

}

default:

{

Console.WriteLine("Invalid choice");

break;

}

}

Console.WriteLine("Do ypu want to continue");

choice = Console.ReadLine();

}

}

//Display(number);

//Insert(5);

//Display(number);

//Insert(70);

//Display(number);

//Insert(25);

//Display(number);

//Delete(60);

//Display(number);

static int MainMenu()

{

int ch;

Console.WriteLine("Main Menu");

Console.WriteLine("1. Insert");

Console.WriteLine("2. Delete");

Console.WriteLine("3. Display");

Console.WriteLine("Enter Choice");

ch = int.Parse(Console.ReadLine());

return ch;

}

static void Display(int[] number)

{

foreach (int num in number)

{

Console.WriteLine(num);

}

}

static void Delete(int num)

{

for (int j = 0; j < Get0(); j++)

{

if (num == number[j])

{

for (int i = j + 1; i < Get0(); i++)

{

number[i - 1] = number[i];

}

number[Get0() - 1] = 0;

}

else if(num == number[Get0() -1])

{

number[Get0() - 1] = 0;

}

}

}

static void Insert(int num)

{

if (num <= number[0])

{

int pos = Get0();

for (int i = pos - 1; i >= 0; i--)

{

number[i + 1] = number[i];

}

number[0] = num;

}

else if (num >= number[Get0() - 1])

{

number[Get0()] = num;

}

else

{

for (int i = 0; i < Get0(); i++)

{

if (num > number[i] && num <= number[i + 1])

{

for (int j = Get0() - 1; j > i; j--)

{

number[j + 1] = number[j];

}

number[i + 1] = num;

break;

}

}

}

}

static int Get0()

{

int pos = 0;

for(int i=0;i<number.Length;i++)

{

if (number[i] == 0)

{

pos = i;

break;

}

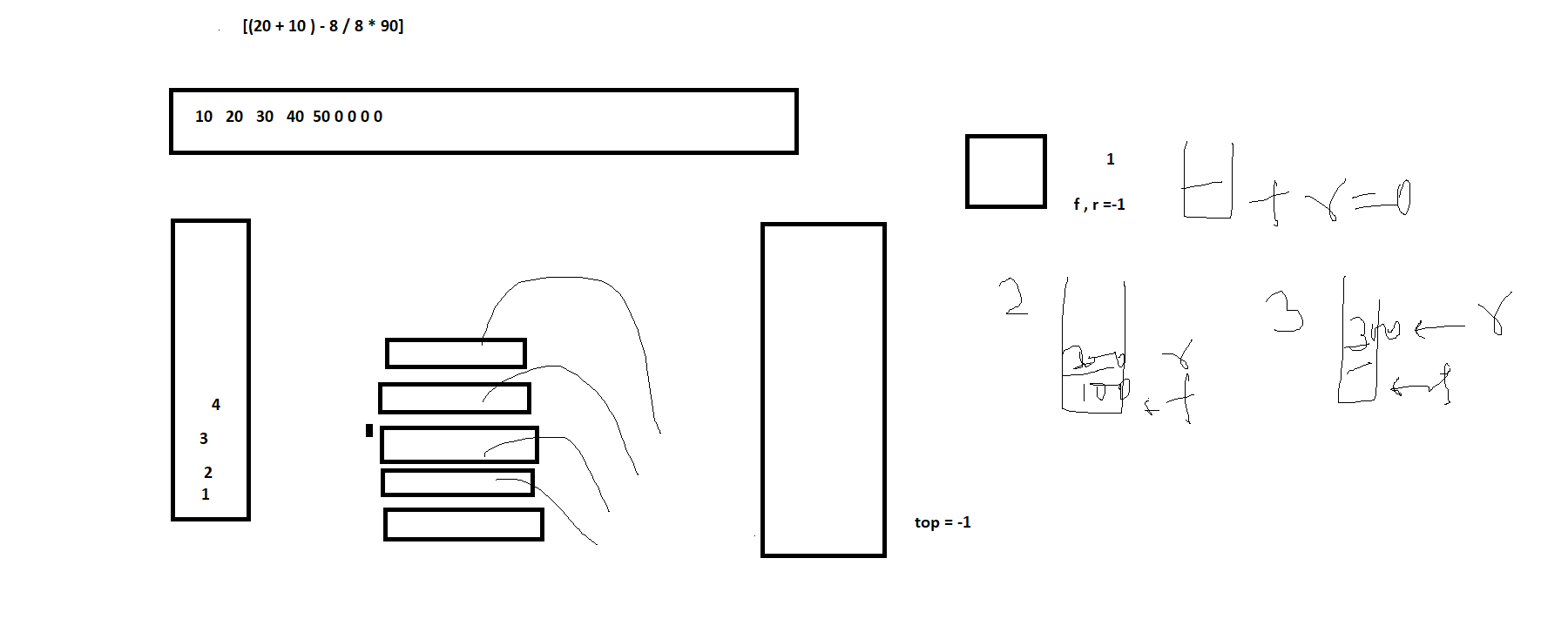
}

return pos;

}

}

}



Stack is a structure which follows LIFO

Circular Queue

