R.M.K College Of Engineering and Technology

Department of Computer Science and Engineering

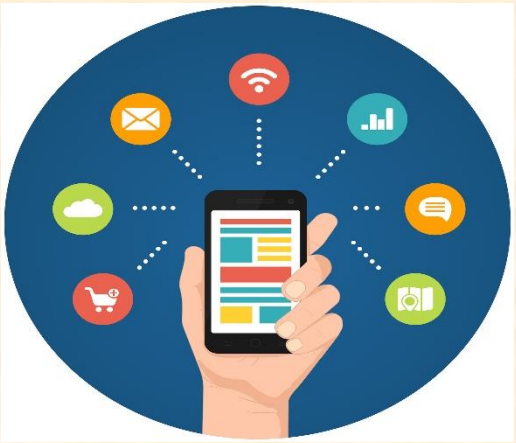
DATE : 18th May 2020 **Targeted Audience** II Year Students	**Facilator** Ms. Abinaya P S, III CSE, Amazon intern	**Convenor :** **Dr.D.Paulraj, Prof. Head CSE Department** **Co-Ordinators:** Dr.M.Vigilson Prem, Prof. CSE Ms.S.Indra Priyadharshini CSE

How to Become a Confident Coder?

Why Coding?

1. You Can Create Anything You Want

Computer programming gives you the ability to digitize your ideas. Imagine being able to actually implement any idea that you could think about! You know that idea you have for an awesome app that would go viral once it was released out into the wild.

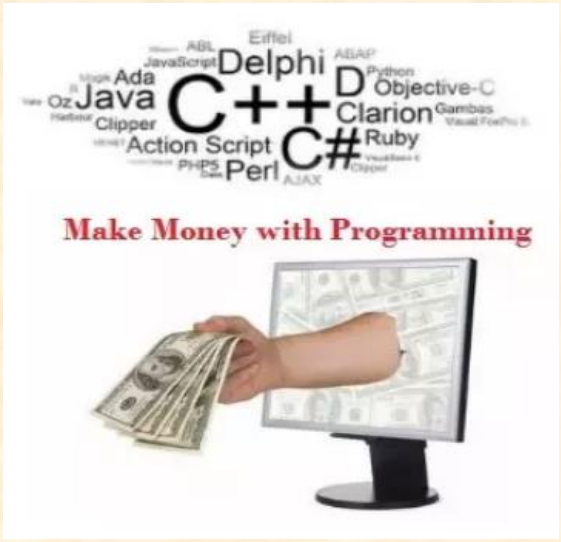


2. Your Job is Guaranteed

We are living in the digital age and the growth of technology does not seem to be coming to a stop. As a matter of fact, the use of electronic devices is growing exponentially every year and so you can be sure that there is going to be no shortage of jobs in the computer programming industry.

3. Make Money

"You can have everything in life that you want, if you will just help enough other people get what they want". – Zig Ziglar



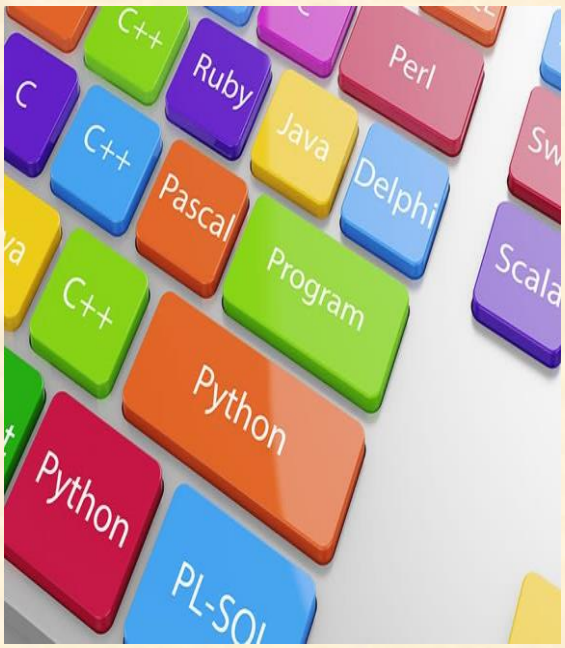
The median salary of a computer programmer stands at $47 per hour which means $97,000 per year!. Statistically, the median is the most robust determinant for averages and if that is coming out to be $47 an hour, you have a very high chance of making that much money and you will also have the opportunity available for higher compensation than the median.

Which Programming Language Do I Choose

The most popular programming languages in use today. There are an incredible number of computer programming languages that are used by coders, software developers, web developers, and

other computer professionals. Here are the common languages we keep hearing:

It is a common misconception that one language is powerful that the other. Any general-purpose language can be used to solve the vast majority of coding problems. From a computational point of view all programming languages solve the same set of problems. But some programming languages are specialized toward certain tasks. So Choose the language which is easiest for the task.



An expect to need a lot of components from libraries (e.g. signal processing, or statistics, or symbolic math), pick a language you're comfortable with that has rich libraries in the relevant areas. For numerical computation and ML, you'll likely find lots of support in Python, but it's certainly not the only option. For embedded systems, you'll naturally get pulled towards C, but it's certainly not the only option.

"Truth: All languages are efficient in their own ways and it's the nature of problem solving that matters."

The Art of Debugging

1.Train Your Eyes 👀

Understand how to look at code and see what it is doing is to step through code you're not familiar with. " Step through and consider each statement. Don't look at the statements as sentences, but get back to your programming roots and see a set of symbols to the left of the equal sign and a set of symbols to the right of the equal sign. You'll be surprised how this can help you hone in quickly to a wrong or duplicate assignment.

2. Get Back to the Basics 🔎

Understanding the behavior of an application to identify the root cause of a bug. That will get the application back on track as quickly as possible.

3. Don't Trust the Documentation 🔐

The documentation is sparse, in flux, and often wrong. Always try to debug by yourself instead of trusting the documents.

4.Practice Makes Perfect 🏋️

A great exercise is to download an open source project, preferably a utility or tool. Spend some time reviewing the source code to determine how you think the code will behave. Then, fire up the debugger and step through each statement. Leave no stone unturned! You may be surprised at what you find and learn. The more often you do this, the better prepared you become.

1.The Best Debug Tool is Your Mind 

Every debugging session should start with a logical walkthrough of the code. You should analyze what you expect it to, and walk through it virtually... *if I pass this here, I'll get that, and then that goes there, and this will loop like that*... this exercise will do more than help you comprehend the code.

a small Easter egg hunt!

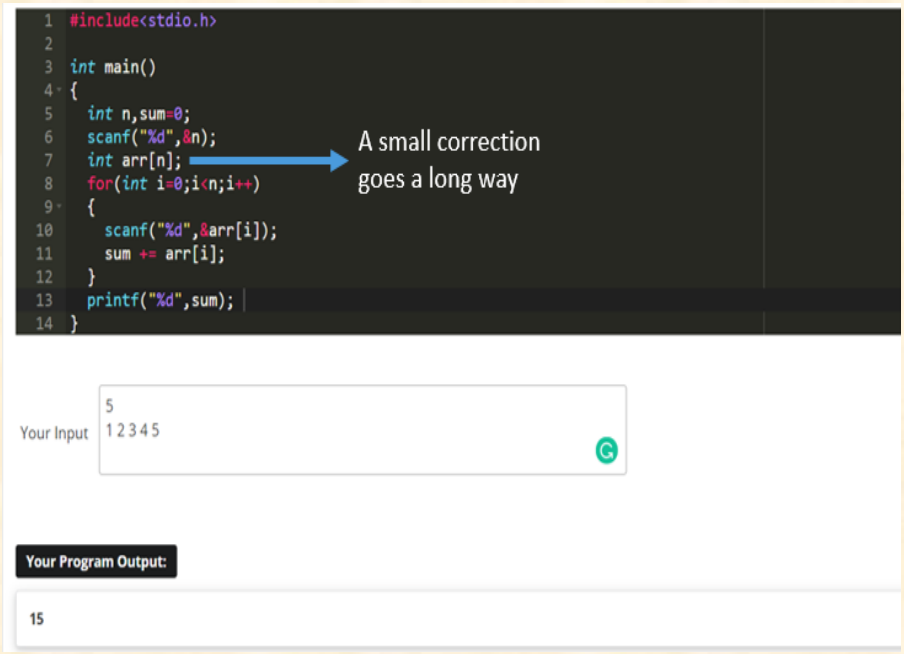
Guess the Output

```c
#include<stdio.h>

int main()
{
   int n,arr[n],sum;
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      scanf("%d",arr[i]);
      sum += arr[i];
   }
   printf("%d",sum);
}
```

Your Input

```
5
1 2 3 4 5
```



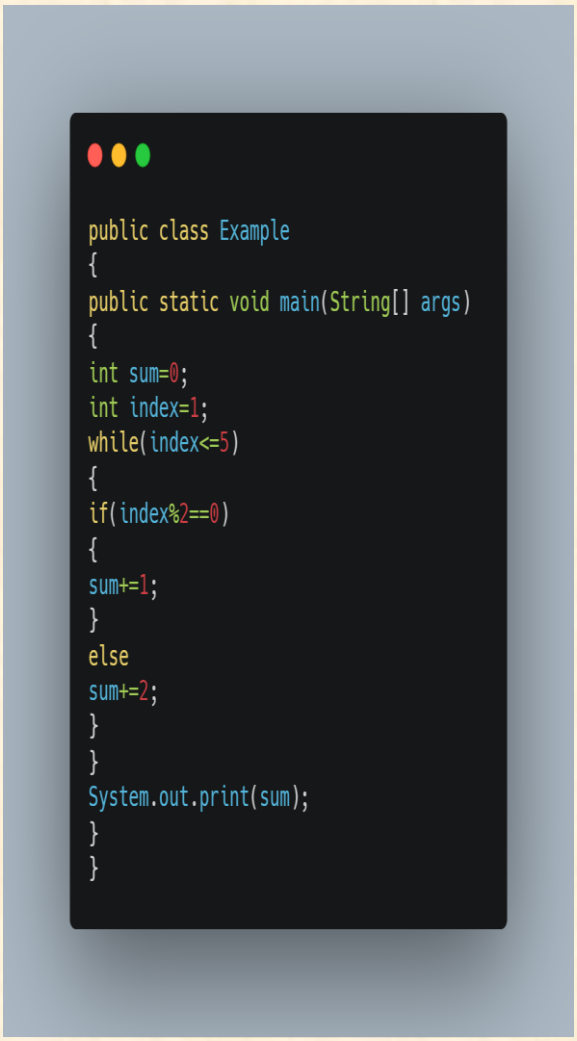
Coding to Impress

1. Naming your Variables

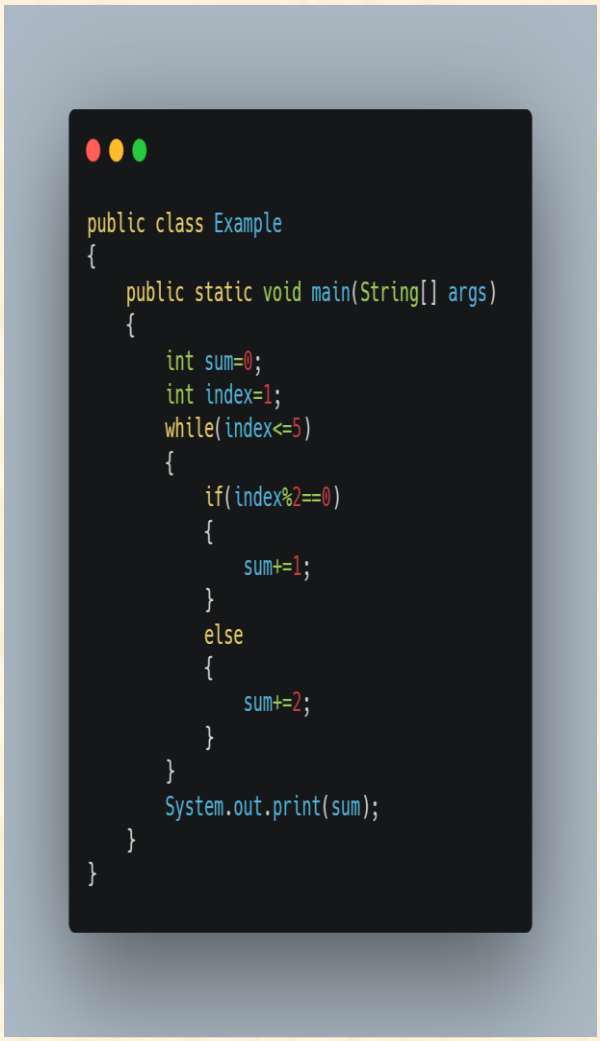
Reasons why naming your variables is very important:

- Helps you understand what you've written
- Easier to debug
- Looks professional
- Practice for real world development

2. The Power of Tabs



```
public class Example
{
public static void main(String[] args)
{
int sum=0;
int index=1;
while(index<=5)
{
if(index%2==0)
{
sum+=1;
}
else
sum+=2;
}
}
System.out.print(sum);
}
}
```

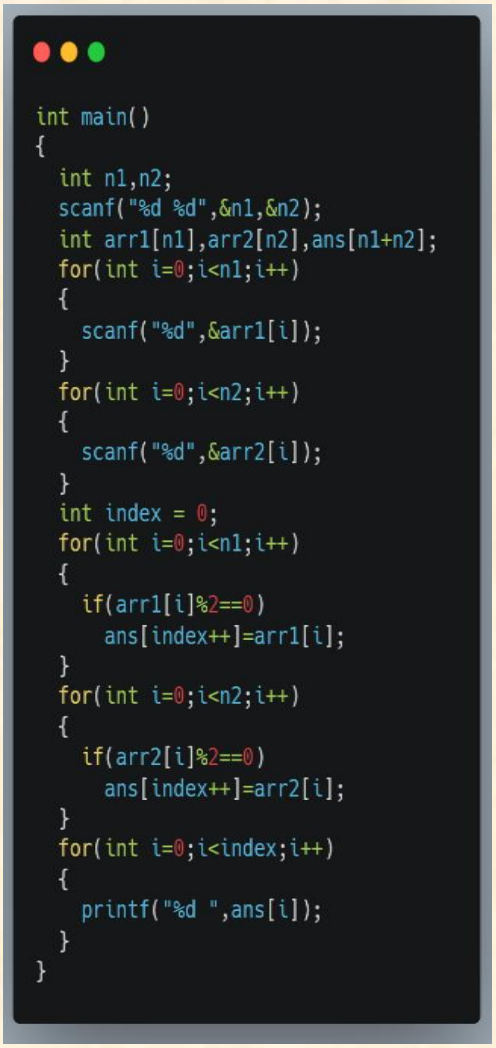


```
public class Example
{
    public static void main(String[] args)
    {
        int sum=0;
        int index=1;
        while(index<=5)
        {
            if(index%2==0)
            {
                sum+=1;
            }
            else
            {
                sum+=2;
            }
        }
        System.out.print(sum);
    }
}
```

3. Modularity

- Functions: The most important part of programming that we often forget
- 4 out of 5 times, coding competitions/rounds will ask you to write only programs
- Platforms like Hackerrank, GeeksforGeeks etc.
- Again, makes you look professional
- Reduces lines of code
- Easy to debug
- Makes coding for Data Structures easy

IMPORTANCE OF FUNCTION



```c
int main()
{
  int n1,n2;
  scanf("%d %d",&n1,&n2);
  int arr1[n1],arr2[n2],ans[n1+n2];
  for(int i=0;i<n1;i++)
  {
    scanf("%d",&arr1[i]);
  }
  for(int i=0;i<n2;i++)
  {
    scanf("%d",&arr2[i]);
  }
  int index = 0;
  for(int i=0;i<n1;i++)
  {
    if(arr1[i]%2==0)
      ans[index++]=arr1[i];
  }
  for(int i=0;i<n2;i++)
  {
    if(arr2[i]%2==0)
      ans[index++]=arr2[i];
  }
  for(int i=0;i<index;i++)
  {
    printf("%d ",ans[i]);
  }
}
```

Comparison of languages which is Known

C	PYTHON	JAVA
Structural, proper and elaborative	Concise and easy to handle	Proper but extends to be easier
Must be hardcoded most of the time	So many libraries to help with many functions	Has plenty of pre-defined classes to help
Excellent choice for beginners to actually learn programming	Suited for learners who want to explore and learn quickly	Best suited for programmers who want to learn concepts in depth
Best language to understand data structures and their working	Not very suitable for data structures due to its flexibility	Suitable for data structure if you're already familiar with the language
Problem solving abilities begin to grow when you code in C	Thirst to explore various new age concepts can be quenched with Python	Java helps you get familiarized with fundamental concepts that form the backbone of IT companies
Use C if you want to improve problem solving	Use Python for projects and exploration	Use Java to learn fundamental concepts