



Compiler Construction

```
static bool init(CURL *conn, char *url, Context *context)
{
    CURLcode code;

    conn = curl_easy_init();

    if (conn == NULL)
    {
        fprintf(stderr, "Failed to create CURL connection\n");
        exit(EXIT_FAILURE);
    }

    code = curl_easy_setopt(conn, CURLOPT_ERRORBUFFER,
        errorBuffer);
    if (code != CURLE_OK)
    {
        fprintf(stderr, "Failed to set error buffer [%d]\n",
            code);
        return false;
    }

    code = curl_easy_setopt(conn, CURLOPT_URL, url);
    if (code != CURLE_OK)
    {
        fprintf(stderr, "Failed to set URL [%s]\n", errorBuffer);
        return false;
    }

    code = curl_easy_setopt(conn, CURLOPT_FOLLOWLOCATION,
        1L);
    if (code != CURLE_OK)
    {
        fprintf(stderr, "Failed to set redirect option [%s]\n",
            errorBuffer);
        return false;
    }

    code = curl_easy_setopt(conn, CURLOPT_WRITEFUNCTION,
        writer);
    if (code != CURLE_OK)
    {
        fprintf(stderr, "Failed to set writer [%s]\n",
            errorBuffer);
        return false;
    }

    code = curl_easy_setopt(conn, CURLOPT_WRITEDATA,
        &context);
    if (code != CURLE_OK)
    {
        fprintf(stderr, "Failed to set write data [%s]\n",
            errorBuffer);
        return false;
    }

    return true;
}

static void StartElement(void *voidContext,
    const xmlChar *name,
    const xmlChar **attributes)
{
    Context *context = (Context *)voidContext;

    if (COMPARE((char *)name, "TITLE"))
    {
        context->title = "";
        context->addTitle = true;
    }
    (void) attributes;
}

// libxml end element callback function
// libxml end element callback function
static void EndElement(void *voidContext,
    const xmlChar *name)
{
    Context *context = (Context *)voidContext;

    if (COMPARE((char *)name, "TITLE"))
        context->addTitle = false;
}

// libxml handling helper function
static void handleCharacters(Context *context,
    const xmlChar *chars,
    int length)
{
    if (context->addTitle)
        context->title.append((char *)chars, length);
}

// libxml PCDATA callback function
static void Characters(void *voidContext,
    const xmlChar *chars,
    int length)
{
    Context *context = (Context *)voidContext;

    handleCharacters(context, chars, length);
}

static void cdata(void *voidContext,
    const xmlChar *chars,
    int length)
{
    Context *context = (Context *)voidContext;

    handleCharacters(context, chars, length);
}
```

Presented by:

Krishan Rupani
500067535
R171218057

Neha Singh
500069028
R171218067

Payal Singla
500069630
R171218118

Under the guidance of:

Dr. Hitesh Kumar Sharma
Assistance Professor
Department of Cybernetics,
School of Computer Science

Introduction:

- Compiler is not a unknown word to all of us, compiler is an integral part of our programming journey.
- It converts our program written in any language to machine understandable code.
- So a programmer writes the code in the compiler and compiler produces output according to the code or it produces error if the code is incorrect in anyway.
- We are making a online web based compiler which will contain most of the language, in future and will reduce the trouble of installing many different compilers on your PC.
- Inspiration to this project came from my own suffering, I have a PC with good computational power but because I do a lot stuff managing so many different compilers for every language made my life really tough.
- Another problem that I have noticed is many edtech startup wishes to have a in build compiler on their web application, we are trying to solve that problem.

Objective:

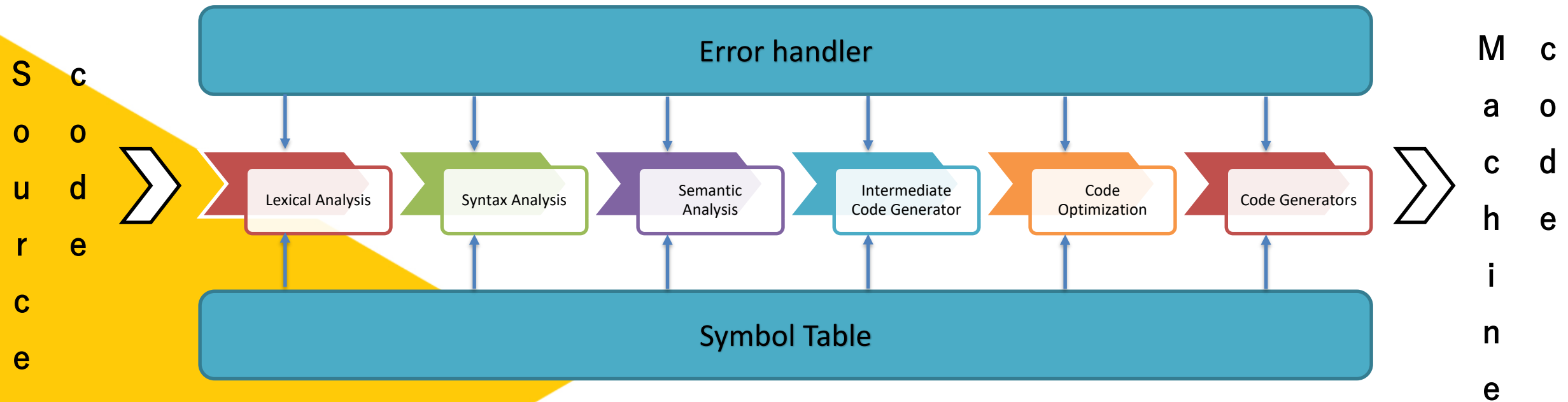
Our aim for this minor project 1 is:

To create a web based compiler for subset of C language.



How Compiler Works:

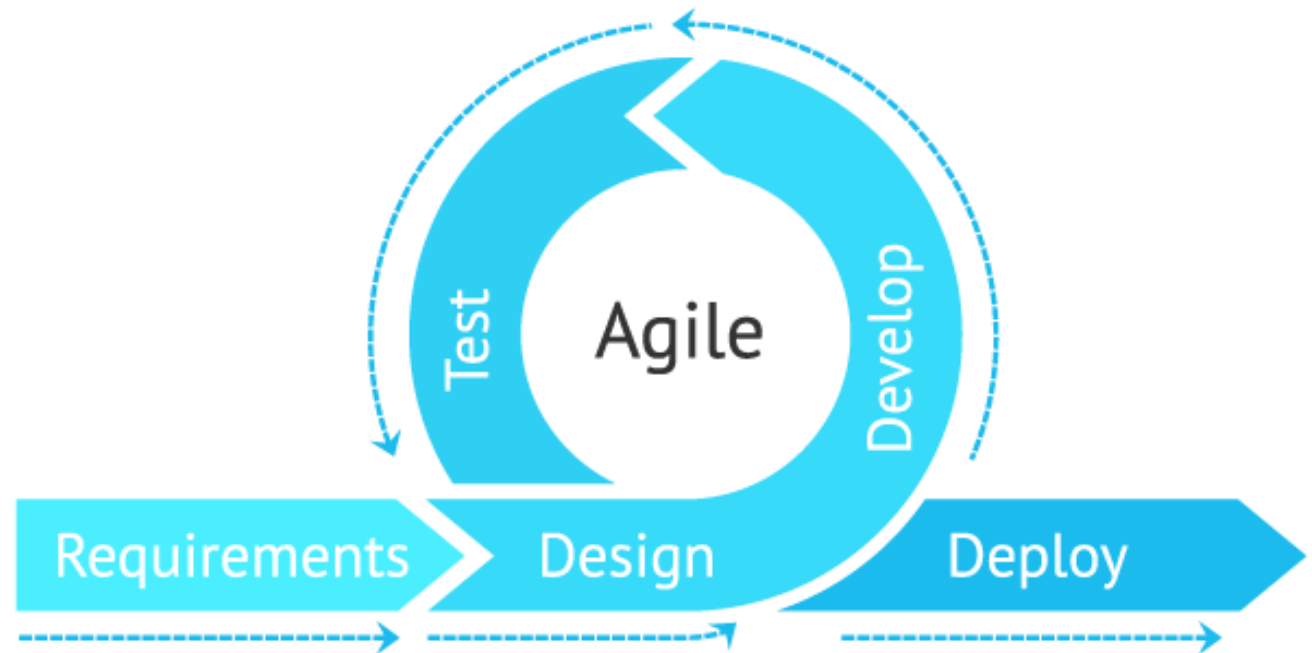
In order to convert the source code in machine understandable code, the source code has to pass 6 phases.



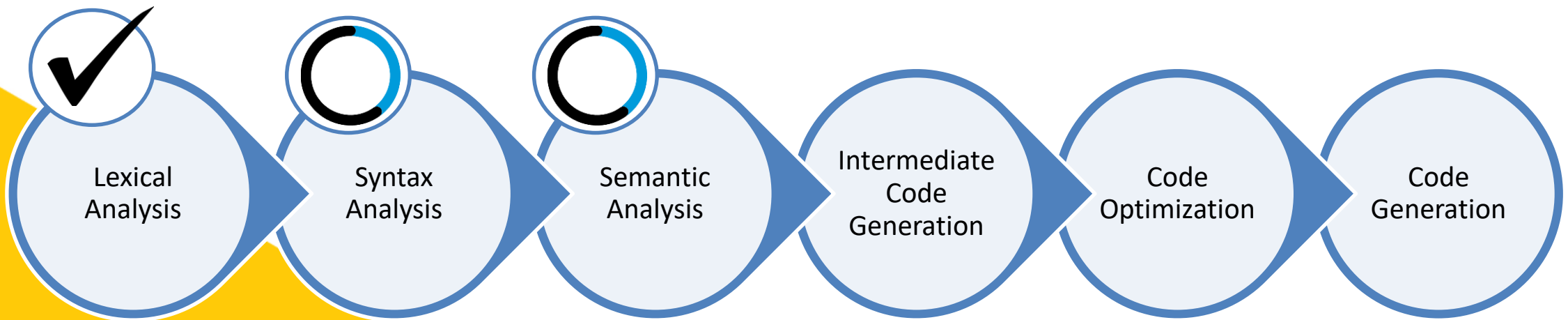
Methodology:

We will make our project using agile methodologies following agile values and principals.

Agile project management is a methodology that is commonly used to deliver complex projects due to its adaptiveness. It emphasizes collaboration, flexibility, continuous improvement, and high quality results.



Progress:



Lexical Analysis(Algorithm):

Main.c

1. Start
2. Give file input
3. Store Tokens
4. Define global variables
5. if argc != 2
 usage(argv[0]);
 A. Print out a usage if started incorrectly
6. Initialise global variables
7. Open input file
 A. Print error if unable to open properly
 B. Infile = fopen(argv[1], "r") == NULL

8. Scan file:
 A. Define tokens: "+", "-", "*", "/",
 "intlit"
 B. while (scan(&T))
 a. Print token string
 C. If token == int
 Print value
9. Finish

Lexical Analysis(Algorithm):

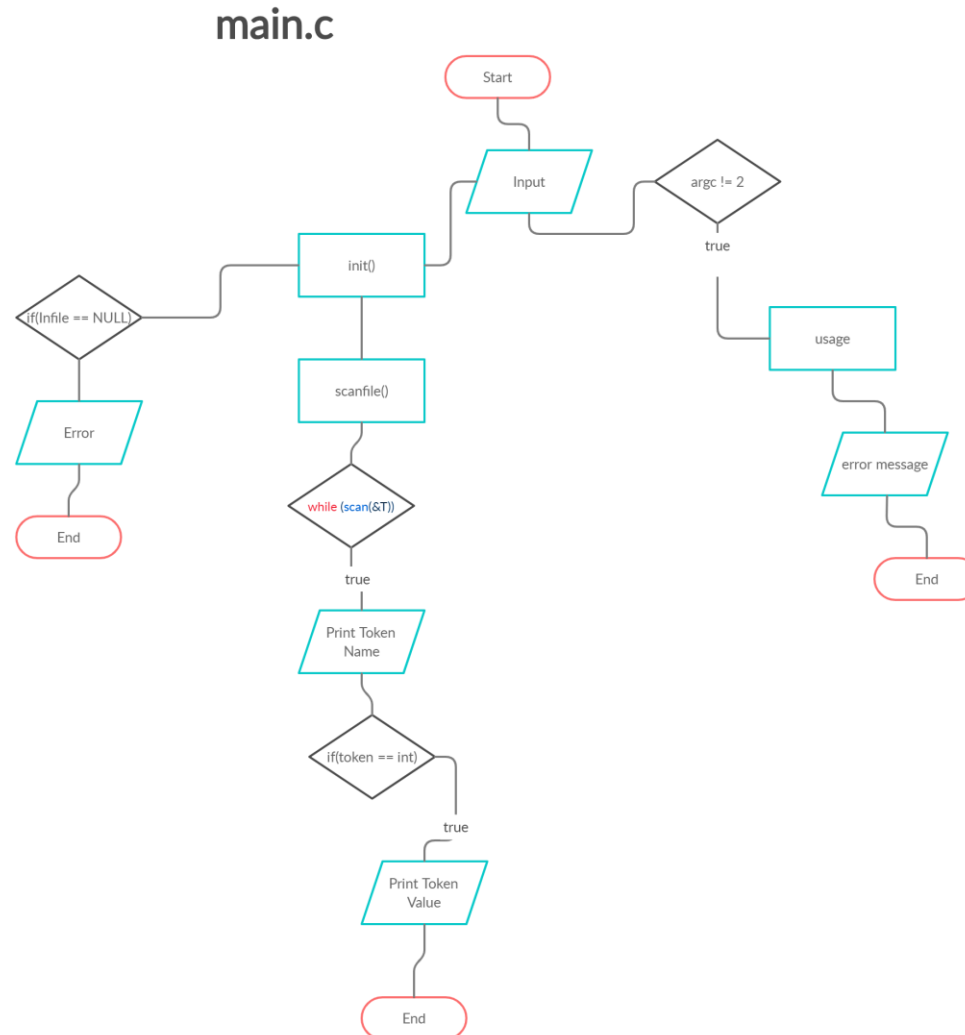
Scan.c

1. Define pointers
2. Define next()
 - a. Define flag variable
 - b. If there is a putback variable
 - i. flag = Putback;
 - ii. Putback = 0;
 - c. Read Input file
 - d. Increase line count
3. Skip unwanted variables
 - a. Define flag variable
 - b. Call next()
 - c. While flag == ' ' or flag == '\t' or flag == '\n' or flag == '\r' or flag == '\f'
 - i. flag = next()

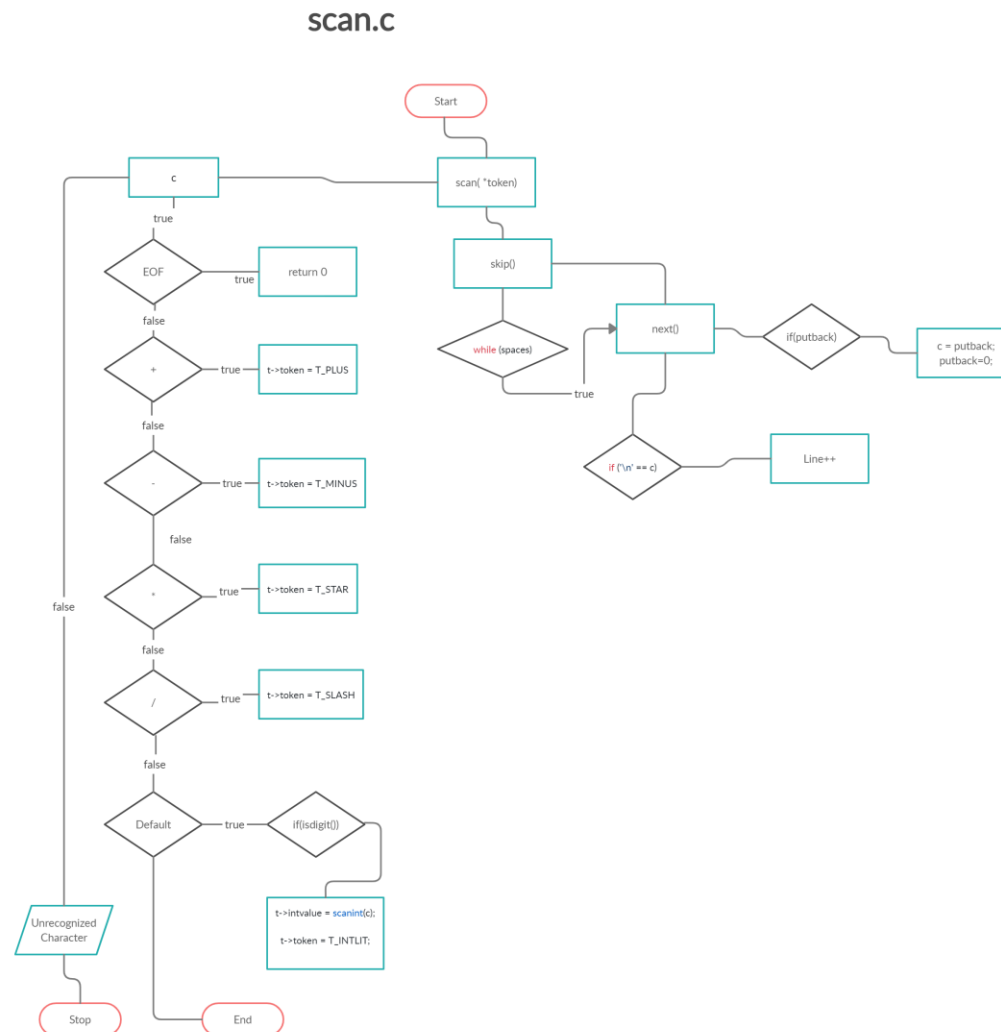
4. Read tokens
 - a. Define switch case:
 - i. Case + : token = T_PLUS
 - ii. Case - : token = T_MINUS
 - iii. Case * : token = T_STAR
 - iv. Case / : token = T_SLASH
 - v. Default: scan integer:
 1. Define variables k, val=0
 2. Convert character to integer:
 - a. While (k = chrpos("0123456789", c)) >= 0
 - i. val = val * 10 + k
 3. If a non integer value is found
 - a. putback()

5. Return token
6. Finish

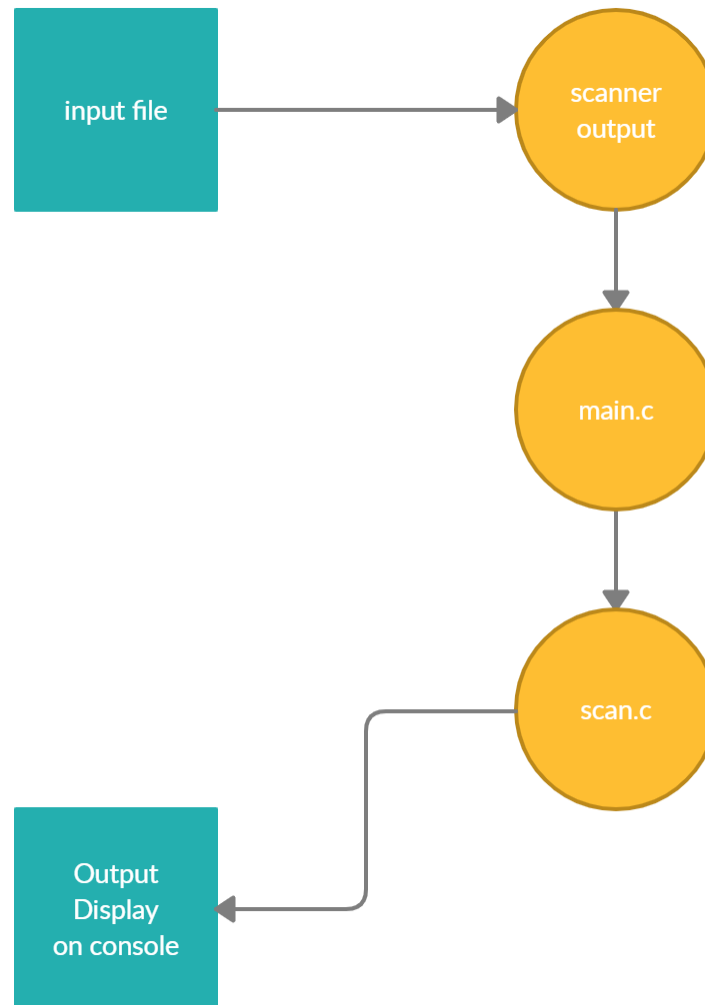
Lexical Analysis(Flowchart):



Lexical Analysis(Flowchart):



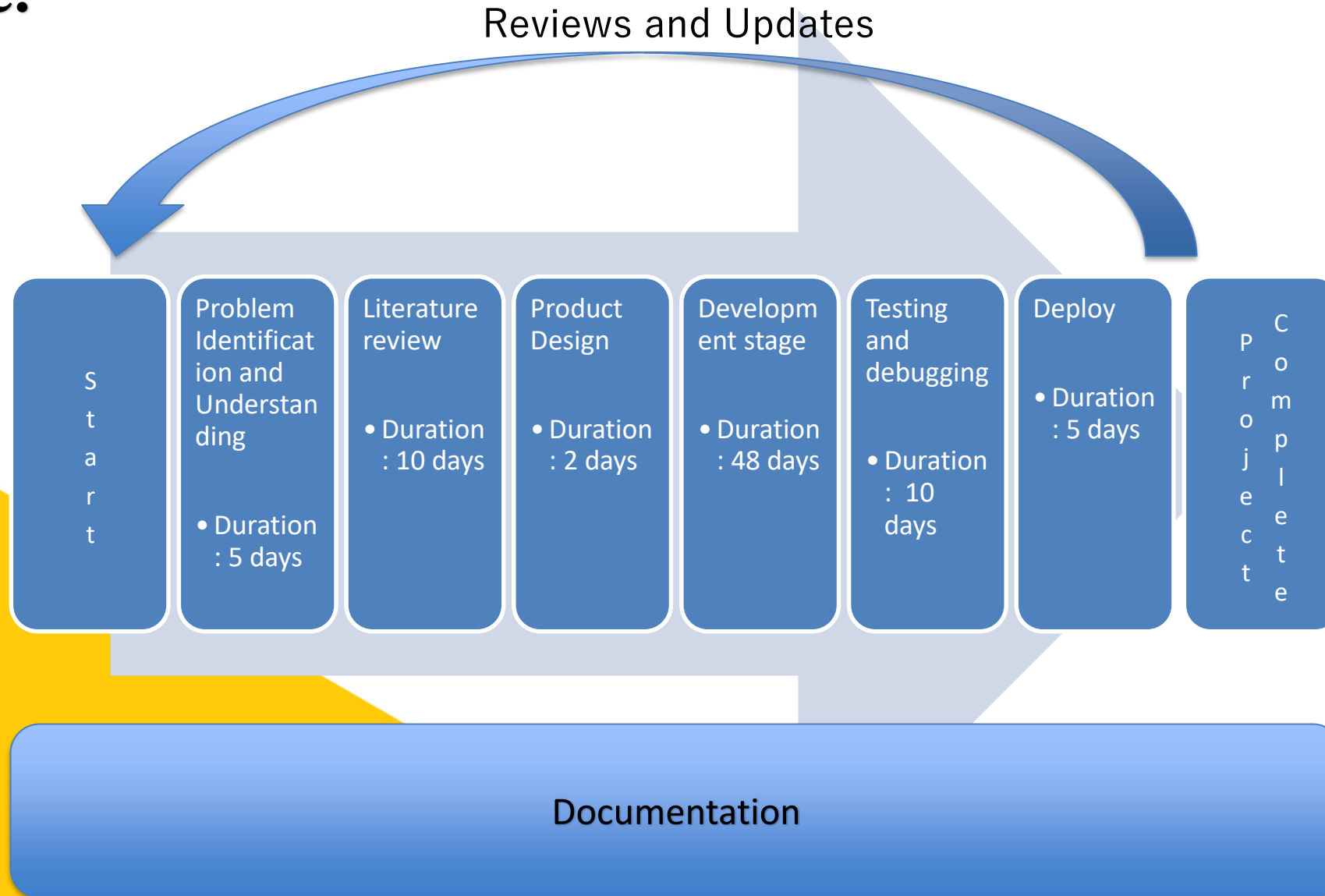
Lexical Analysis:



Lexical Analysis(Output):

```
neha@neha-VirtualBox:~/minor1/Web-based-compiler/Lexical Analyser$ ls
data.h decl.h defs.h InputFile1 main.c Makefile scan.c
neha@neha-VirtualBox:~/minor1/Web-based-compiler/Lexical Analyser$ make
cc -o scanner -g main.c scan.c
neha@neha-VirtualBox:~/minor1/Web-based-compiler/Lexical Analyser$ ls
data.h decl.h defs.h InputFile1 main.c Makefile scan.c scanner
neha@neha-VirtualBox:~/minor1/Web-based-compiler/Lexical Analyser$ cat InputFile1
2
Text Editor
neha@neha-VirtualBox:~/minor1/Web-based-compiler/Lexical Analyser$ ./scanner InputFile1
Token intlit, value 2
Token /
Token intlit, value 3
Token *
Token intlit, value 5
Token +
Token intlit, value 7
Token -
Token intlit, value 11
neha@neha-VirtualBox:~/minor1/Web-based-compiler/Lexical Analyser$ ls
data.h decl.h defs.h InputFile1 main.c Makefile scan.c scanner
neha@neha-VirtualBox:~/minor1/Web-based-compiler/Lexical Analyser$ make clean
rm -f scanner *.o
neha@neha-VirtualBox:~/minor1/Web-based-compiler/Lexical Analyser$ ls
data.h decl.h defs.h InputFile1 main.c Makefile scan.c
neha@neha-VirtualBox:~/minor1/Web-based-compiler/Lexical Analyser$
```

Schedule:



THANK YOU



UNIVERSITY WITH A PURPOSE