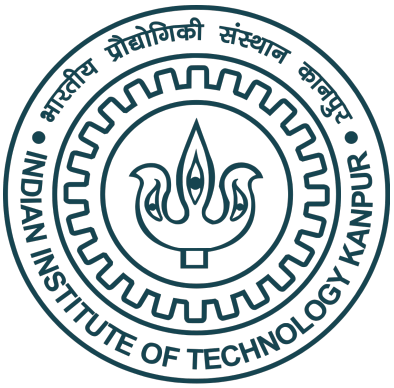
ANAA C++ Compiler Specifications

January 24, 2022



Course CS335A (Compiler Design)

Instructor Dr. Amey Karkare (Professor) CSE Dept. IIT Kanpur (karkare@iitk.ac.in)

Dr. Subhajit Roy (Professor) CSE Dept. IIT Kanpur (subhajit@iitk.ac.in)

Semester 2021-22-II (Jan - Apr 2022)

Participants

Roll No	Name	Email
190185	Aryan Kumar	aryankmr@iitk.ac.in
190535	Naveen Kumar Mathur	naveenkm@iitk.ac.in
190082	Akash Kumar Bhoi	akashkb@iitk.ac.in
190802	Shinde Ankit Jagdish	ankits@iitk.ac.in

Contents

1 Introduction

Source (S): C++ **Implementation (I): C** **Target (T): x86**

For implementation we are using C . Our source language is C++ and our target is x86.

Repository Link: https://github.com/ankits2511/ANAA.

STRUCTURE OF REPOSITORY:

`README.md` : Details of our project and the steps to build and run it.

`Makefile` : To build the implementation.

`bin` : Binaries/objects stored here.

`docs` : Contains documents related to specifications(features of compiler).

`src` : All the source code here.

`tests` : Test cases stored here.

2 Features of ANAA C++ Compiler

Our compiler is incorporated with the following features:-

2.1 Native Data Types

2.1.1 Integer

- Keyword used for integer data types is int. Integers typically requires 4 bytes of memory space and ranges from -2,147,483,648 to 2,147,483,647.

2.1.2 Character

- Character data type is used for storing characters. Keyword used for character data type is char. Characters typically requires 1 byte of memory space and ranges from -128 to 127 or 0 to 255.

2.1.3 Boolean

- Boolean data type is used for storing boolean or logical values. A boolean variable can store either true or false. Keyword used for boolean data type is bool.

2.1.4 Float

- Floating Point data type is used for storing single precision floating point values or decimal values. Keyword used for floating point data type is float. Float variables typically requires 4 byte of memory space.

2.1.5 Double Float

- Double Floating Point data type is used for storing double precision floating point values or decimal values. Keyword used for double floating point data type is double. Double variables typically requires 8 byte of memory space.

2.1.6 void

- Void means without any value. void datatype represents a valueless entity. Void data type is used for those function which does not returns a value.

data type	size(Bytes)	range
short int	2	-32,768 to 32,768
unsigned short int	2	0 to 65,535
int	4	-2,147,483,648 to 2,147,483,647
unsigned int	4	0 to 4,294,967,295
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
long long int	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	-
double	8	-
long double	12	-

2.2 Variables and Expressions

2.2.1 Variables

- Variables are containers for storing data values.
- The syntax for defining variables is
  - data_type variable_name;
  - data_type variable_name, variable_name, ...;
- data_types include:
  - int [32 bit]
  - float [decimal point system]
  - char [stores a character] [surrounded by single quotes]
  - bool [a boolean value (True/False)]
- name of data_types is registered as keywords.
- variable_name convention:
  - variable_name can contain alphabets, numbers and underscore
  - variable_name can't start with numbers.
  - variable_name must contain an alphabet or underscore.
- A variable is a named location in a memory.
- A program can manipulate the data using a variable.
- A variable's location is used to hold the value of the variable.

2.2.2 Expressions

- An expression is a combination of operators, constants and variables. An expression may consist of one or more operands, and zero or more operators to produce a value.

- An expression can also be said to a formula to perform any operation.
- An operand in a expression may be a function reference, an array element, a variable, or any constant.
- The expressions are evaluated by pre-defined precedence of operators.
- **Types of Operators**
  - Assignment Operators { =, +=, -=, *=, /= , %= , &= }
  - Increment-Decrements Operators { ++, −− }
  - Arithmetic Operators { +, -, *, /, , %, &, Γ, <<, >> }
  - $\mathcal{L}\wr\}\rangle\rfloor\dashv\updownarrow\mathcal{O}_{\surd}\rceil\nabla\dashv\sqcup\wr\nabla\int\{\searrow\searrow\Leftrightarrow||\Leftrightarrow\rightarrow\}$
  - $\mathcal{C}\wr\Updownarrow_{\surd}\dashv\nabla\rangle\int\wr\backslash\int\mathcal{O}_{\surd}\rceil\nabla\dashv\sqcup\wr\nabla\int\{==,!=,<,>,<=,>=\}$
  - $\mathcal{M}\rceil\Updownarrow\lfloor\rceil\nabla\dashv\rfloor\rfloor\rceil\int\int\mathcal{O}_{\surd}\rceil\nabla\dashv\sqcup\wr\nabla\int\{a[b]\Leftrightarrow a.b\Leftrightarrow a->b\Leftrightarrow *a\Leftrightarrow \&a\}$
- Based upon number of operand it may classify to Unary, Binary, Ternary operator.

2.3 Control Structure

2.3.1 Conditional

- ***if***
  Syntax:

```
if(condition)
  {Body...}
```

  If condition found true, the body part will execute.

- ***if − else***
  Syntax:

```
if(condition)
  {Body-1...}
else
  {Body-2...}
```

  If condition found true then code of Body-1 will execute otherwise code inside Body-2 will execute.

- ***Nested if − else***
  Syntax:

```
if(condition)
  {
  Body-1...
    if(condition-2){
        ...
```

```
    }
  }
else
  {Body-2...
   if(condition-3)
     {Body-3...}
   else
     {Body-4...}
  }
```

It is if-else condition under if else.

2.3.2 Loops

- ***for***
  Syntax:

```
for(initialization; condition; increment-decrements;){
Body...
}
```

  - Until the condition holds it will execute the code inside the body part of the loop.
  - Note that the initialization might be done before the loop.

- ***while***
  Syntax:

```
while(condition){
    Body...
}
```

  - Until the condition holds it will execute the code in the body part of while loop.

- ***do − while***
  Syntax:

```
do{
    Body...
}while(condition)
```

  - It will the run the code once then check the condition then if it holds it will run again.

- ***switch − case***
  Syntax:

```
switch(val){
    case a:
        break;
    case b:
        .
```

```
                .
                .
            default:
                ....
        }
```

- It will match the val(generally numerical value) and execute the particular block.
- It stop executing until it hits break statement.
- default is the case when no cases matches.

2.4 [Input/Output] Statements

2.4.1 cin

- It is a predefined variable that reads data with the operator (>>).
- It reads the data from standard input stream for the program.
- Can add as many cin objects as we want to read.

2.4.2 cout

- It is a predefined variable that output data with the operator (<<).
- It outputs the data to standard output stream for the program.
- Can add as many cout objects as we want to print.
- It does not insert a new line at the end of the output.

2.4.3 Streams

The connection between a program and a data source or destination is called a stream.

- Input Stream:
  - An input stream handles data flowing into a program.
  - If the direction of flow of bytes is from the device(for example, Keyboard) to the main memory.
- Output Stream:
  - An output stream handles data flowing out of a program.
  - If the direction of flow of bytes is opposite, i.e. from main memory to device (display screen).

2.5 Array

An array in C/C++ or be it in any programming language is a collection of similar data items stored at contiguous memory locations and elements can be accessed randomly using indices of an array. They can be used to store collection of primitive data types such as int, float, double, char, etc of any particular type. In C/C++, array elements are indexed beginning at position zero, not one.

Example– You declare an array by specifying the data type for its elements, its name, and the number of elements it can store. Here is an example that declares an array that can store 3 integers:

int a[3] = 1,2,3;

1	2	3

Index of integer 1 = 0
Index of integer 2 = 1
Index of integer 3 = 2

For standard C/C++ code, the number of elements in an array must be positive.

Different ways of declaring an array are as follows:-

- int a[3];

2192	451	13918

- int a[3]=1,2,3;

1	2	3

- int a[3]=;

0	0	0

  int a[3]=0;

0	0	0

  int a[3]=1;

1	0	0

- int a[3]=[0.. 1]=3;

3	3	0

  int a[]=[0.. 1]=3;

3	3

- int *a; or int* a; or int * a; or int*a;

2.6 Functions

Functions are C++ entities that associate a sequence of statements with a name and a list of zero or more parameters.

Function definition syntax :-

```
return_type function_name(arg_1, arg_2, ...., arg_n){
    Function body..
    }
```

Call to function syntax :-

```
function_name(arg_1,arg_2,...,arg_n);
```

2.7 User defined types(class/struct)

2.7.1 Class

- A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package.
- The data and functions within a class are called members of the class.
- Defining a class is same as defining a blueprint for a data_type. **Declaration :**
  - A class definition starts with the keyword class followed by:
    - the class name
    - and the class body,
    - enclosed by a pair of curly braces.
- A class definition must be followed either by a semicolon or a list of declarations.

Example Declaration:

```
class Demo_Class {
   private:
                .
                .
                .
                .

   public:
      data_type Variable_name1, Variable_name2, . . .;
      return_type function_name(arg_1, arg_2,...,arg_n){
            Body...
      }
            .
            .
            .
};
```

- Keyword public determines the access attributes of the members of the class.
- Public members can be accessed from outside the class anywhere within the scope, and can be accessed using the direct member access operator (.)
- Private and protected members can not be accessed directly using operator (.)

Features of a Class:

- Class Member Functions:
  - It is a function that has its definition or its prototype within the class definition like any other variable.
- Class Access Modifiers:
  - A class member can be defined as public, private or protected.
  - By default members would be assumed as private.

- Constructor:
  - A class constructor is a special function in a class that is called when a new object of the class is created.
- Destructor:
  - A destructor is also a special function which is called when created object is deleted.
- Static Members of a Class:
  - Both data members and function members of a class can be declared as static.

2.7.2 Structures

- A STRUCT is a data structure that can be used to store together elements of different data_types.
- A structure is a user-defined data type.
- The structure creates a data type for grouping items of different data types under a single data type.
- A Struct is a collection of different data types and it is similar to the class that holds different types of data.
- If access specifier is not declared explicitly, then it is public by default.

Example declaration of a struct :

```
Struct Demo_Struct_name{
        data_type_1 Variable_name_1;
        data_type_2 Variable_name_2;
        data_type_3 Variable_name_3;
        data_type_4 Variable_name_4;
};
```

2.8 Pointers

- Pointer is a variable whose value is the address of another variable.
- Pointers can be used extensively for these purposes:
  - to allocate new objects on the memory (heap).
  - To pass functions to other functions
  - To iterate over elements in arrays
- The general form of a pointer variable declaration is:

  data_type *variable_name;
  where [data_type can be bool, int, float, char]

  - Here, data_type is the Pointer's base type [a valid type].
  - Asterisk is being used to designate a variable as a pointer.

- The only difference b/w pointers of different data_types is the data type of the variable or constant that the pointer points to.

2.8.1 Features of Pointers

- Null Pointer: is a constant with a value of zero defined in standard.
- Pointer Arithmetic:
  - Four arithmetic operators that can be used on pointer are [++, -- , +, -].
- Pointers and Arrays have a close relationship.
- Passing Pointers to Functions:
  - Passing an argument by reference or by address both enable the passed argument to be changed in the calling function by the called function.
- Return Pointer from Functions:
  - Our language allows a function to return a pointer to local variable, static variable and dynamically allocated memory as well.

2.9 Library Functions [Strings/Math]

Library functions are built-in functions that are grouped together and placed in a common location called library.Each function here performs a specific operation. We can use this library functions to get the pre-defined output.
Some C++ library functions and their descriptions are as follows :

- **String functions (< *cstring* >):**
  - Contains function prototypes for C-style string-processing functions. This header file replaces header file < *string.h* >.
- **Math functions (< *cmath* >):**
  - Contains function prototypes for math library functions. This header file replaces header file < *math.h* >. Math functions includes the functions for sin(x), cos(x), tan(x), asin(x), acos(x), exp(x), log(x), log 10(x), sqrt(x), pow(x,y), abs(x), fabs(x)

3 Table of Abbreviations (Glossary)

Abbrev.	Description	Similar
ANAA	**Ankit Naveen Aryan Akash**	
x86	**A family of instruction set architectures initially developed by Intel based on the Intel 8086 microprocessor**	
Repo.	**Repository**	
STRUCT	**Structure**	

References

1.https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html
2.https://gcc.gnu.org/onlinedocs/gcc.pdf