[Samsung PRISM] January Review Report

Grammar Based C/C++ to Rust Transpiler

Team

1. College Professor(s): Dr. Srinivas Pinisetty
2. Students:
    1. Mithun Chandrashekar
    2. Arnav Kumar Behera
    3. Vedanta Mohapatra
3. Department: Computer Science and Engineering

Date: 19 Dec 2023

Work-let Name: Grammar Based C/C++ to Rust Transpiler

Worklet Details

1. Worklet ID: 23SE09
2. College Name: IIT BBS

KPIs achieved till now

- Added support for multidimensional arrays to be passed into functions by reference (avoid code like *(ptr+i) and instead use ptr[i])
- Added translation of pointers in function parameters to unsafe rust

Any Challenges/ Issues faced

- Looking at use case of pointers to decide whether to treat them as arrays or references needs lookahead into the parse tree.
- Converting (a+offset)* to a[offset] makes sense only if offset is non-negative. C allows negative indexing as well as offsets.

Next Steps

- Addition of Trait declaration to the generated Rust code
- Further Work on pointers

Key Achievements/ Outcome till now

Date: 25/01/2024

Experimental Results

- Passing Arrays

test.cpp > ...

```cpp
#include<bits/stdc++.h>
int add(int a[10],size_t n){
    int sum=0;
    for(size_t i=0;i<n;i++){
        sum+=a[i];
    }
    return sum;
}
int main(){
    int x=4;
    int a[10]={1,2,3,4,5,6,7,8,9,10};
    int sum=add(a,10);
    printf("sum is : %d\n",sum);
}
```

test_converted.rs

```rust
#![allow(warnings, unused)]
fn add(mut a: [i32; 10], mut n: usize) -> i32 {
    let mut sum: i32 = 0 as i32;
    let mut i: usize = 0 as usize;
    while i < n {
        sum += a[i];
        i += 1;
    }
    return sum;
}
fn main() {
    let mut x: i32 = 4 as i32;
    let mut a: [i32; 10] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
    let mut sum: i32 = add(a, 10) as i32;
    println!("sum is : {}\n", sum);
}
```

Experimental Results

- Passing Multi-Dimensional Arrays

test.cpp > ...

```cpp
#include<bits/stdc++.h>
void print(int a[3][3],size_t n){
    for(size_t i=0;i<n;i++){
        for(size_t j=0;j<n;j++){
            printf("%d ",a[i][j]);
        }
        printf("\n");
    }
}
int main(){
    int a[3][3];
    print(a,3);
}
```

test_converted.rs

```rust
#![allow(warnings, unused)]
fn print(mut a: [[i32; 3]; 3], mut n: usize) {
    let mut i: usize = 0 as usize;
    while i < n {
        let mut j: usize = 0 as usize;
        while j < n {
            println!("{} ", a[i][j]);
            j += 1;
        }
        println!("\n");
        i += 1;
    }
}
fn main() {
    let mut a: [[i32; 3]; 3];
    print(a, 3);
}
```

Experimental Results

- **Passing pointers (unsafe):**

```c
int cmpfunc(const int *a, const int *b) {
  return (*(a + 2) - *b);
}
```

```rust
#![allow(warnings, unused)]
fn cmpfunc(
    // Handling Pointers...
    a: *mut i32, // Handling Pointers..
    b: *mut i32,
) -> i32 {
    return (*(a + 2) - *b);
}
```