CSE443 Compilers

https://www.cse.buffalo.edu/courses/cse443/

Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall

Department of Computer Science and Engineering

University at Buffalo

Team formation and meeting scheduling

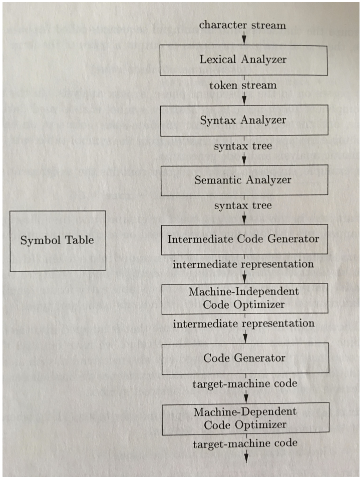
- Form teams as soon as possible, no later than Thursday next week (after add/drop).
- Teams must be of size 3 or 4.
- One member of each team must make a Piazza private post to me with the UBIT and GitHub username of each team member.
- Add code for the team must be maintained in a private git repo hosted on GitHub.
  - I will set these up via GitHub classroom once teams are formed.
  - DO NOT SET UP REPOS ON YOUR OWN BEFORE THEN!

The Compiler

high-level program (source)

$\Downarrow$

low-level program (assembly)



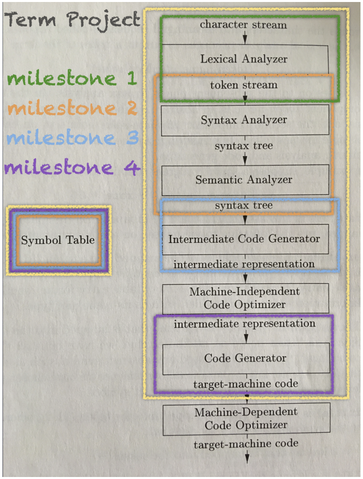
(figure 1.6, page 5 of text)

The Project

high-level program (source)

⇓

low-level program (assembly)



(figure 1.6, page 5 of text)

Deep understanding

identifier

vs

name

vs

variable

STATIC (in program text)

identifier

x

name

y.x

DYNAMIC (at runtime)

variable

location in memory

example 1

```
void foo(void) {
    int x = 0;
    printf(x);
}

void bar(void) {
    double x = 3.8;
    printf(x);
}
```

example 2

```
struct Pair {
    int x;
    int y;
};

void bar(void) {
    struct Pair r, s;
    /* ... */
}
```

example 3

```
int f(int x) {
    if (x == 0) { return 1; }
    else { return x * f(x-1); }
}
```

CODE

RUNTIME

variable

identifier
x

variable

variable

- identifier in distinct scopes
- identifier in distinct record instances
- identifier in distinct function invocations

Order of evalution 1

Does source code completely determine order of evaluation/execution at machine language level?

Order of evaluation 2

```
a + b * c
```

What is the order of evaluation?

Order of evaluation 2

```
a + b * c
```

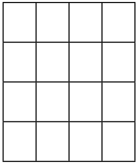
What is the order of evaluation of the expressions?

Order of evaluation 2

```
a + b * c
```

How many expressions are there?

Order of evaluation 2



How many boxes are there?

Order of evaluation 3

```
f() + g() * h()
```

What is the order of evaluation?

What is the order of the function calls?

Must `g` be called before `f`?

Order of evaluation 3

```
f() + f() * f()
```

How many times will `f` be called?

Could it be just once?

If it cannot be just once, is order important?

Order of evaluation 3

```
f() + f() * f()
```

If the value of `f()` depends on mutable persistent state, then the value returned by each call can be different.

Order of evaluation 3

```
f() + f() * f()
```

If `f` is known to be referentially transparent[1] then each call to `f()` will produce the same value.

We can then compute `f()` once and use its value multiple times.

[1] "Referential transparency and referential opacity are properties of parts of computer programs. An expression is called referentially transparent if it can be replaced with its corresponding value without changing the program's behavior. This requires that the expression be pure, that is to say the expression value must be the same for the same inputs and its evaluation must have no side effects. An expression that is not referentially transparent is called referentially opaque." https://en.wikipedia.org/wiki/Referential_transparency

Program semantics

```
#include <stdio.h>

int main() {
  int i = 0;
  int sum = 0;
  while (i <= 10) {
    sum = sum + i;
    printf("sum of integers from 0 to %d is
       %d.\n",i,sum);
    i = i + 1;
  }
}
```

What determine program semantics?

Program semantics

```
#include <stdio.h>

int main() {
  int i = 0;
  int sum = 0;
  while (i <= 10) {
    sum = sum + i;
    printf("sum of integers from 0 to %d is
       %d.\n",i,sum);
    i = i + 1;
  }
}
```

What is the code highlighted in red?

Program semantics

```
#include <stdio.h>

int main() {
  int i = 0;
  int sum = 0;
  while (i <= 10) {
    sum = sum + i;
    printf("sum of integers from 0 to %d is
       %d.\n",i,sum);
    i = i + 1;
  }
}
```

What is the code highlighted in red?

Program semantics

```
- on part d'un entier ;
- s'il est pair, on le divise par 2 ;
- sinon, on le multiplie par 3 et on ajoute 1 ;
- on recommence la même opération sur l'entier obtenu, et ainsi de suite ;
- la suite s'arrête si on arrive à 1. */

syracuse :
  durée est un nombre
  e est un nombre
  début
    e prend 14
    tant que e != 1 lis
      durée prend durée + 1
      si (e mod 2) = 0, e prend e / 2
      sinon e prend e * 3 + 1
      affiche e
    ferme
    affiche "durée = {durée}"
```

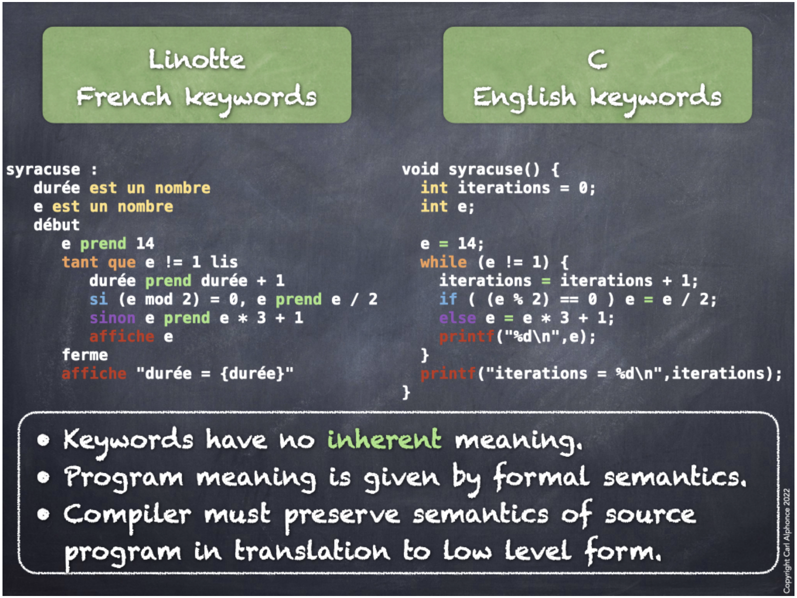
More information: `http://langagelinotte.free.fr/wordpress/`

Program semantics

```
/* The Syracuse sequence is defined as follows:
- it starts with any natural number > 0
- if it is even, we divide by 2
- else we multiply by 3 and add 1
- the process is repeated on the result
- the process ends when the result is 1 */

void syracuse() {
  int iterations;
  int e;

  iterations = 0;
  e = 14;
  while (e != 1) {
    iterations = iterations + 1;
    if ( (e % 2) == 0 ) e = e / 2;
    else e = e * 3 + 1;
    printf("%d\n",e);
  }
  printf("iterations = %d\n",iterations);
}
```



Linotte
French keywords

C
English keywords

```
syracuse :
   durée est un nombre
   e est un nombre
   début
      e prend 14
      tant que e != 1 lis
         durée prend durée + 1
         si (e mod 2) = 0, e prend e / 2
         sinon e prend e * 3 + 1
         affiche e
      ferme
      affiche "durée = {durée}"
```

```
void syracuse() {
  int iterations = 0;
  int e;

  e = 14;
  while (e != 1) {
    iterations = iterations + 1;
    if ( (e % 2) == 0 ) e = e / 2;
    else e = e * 3 + 1;
    printf("%d\n",e);
  }
  printf("iterations = %d\n",iterations);
}
```

- Keywords have no inherent meaning.
- Program meaning is given by formal semantics.
- Compiler must preserve semantics of source program in translation to low level form.

Homework

- form teams and identify potential meeting times
    - follow instructions in Piazza (will be posted over weekend)
- Read sections 2.6 and 3.1

License