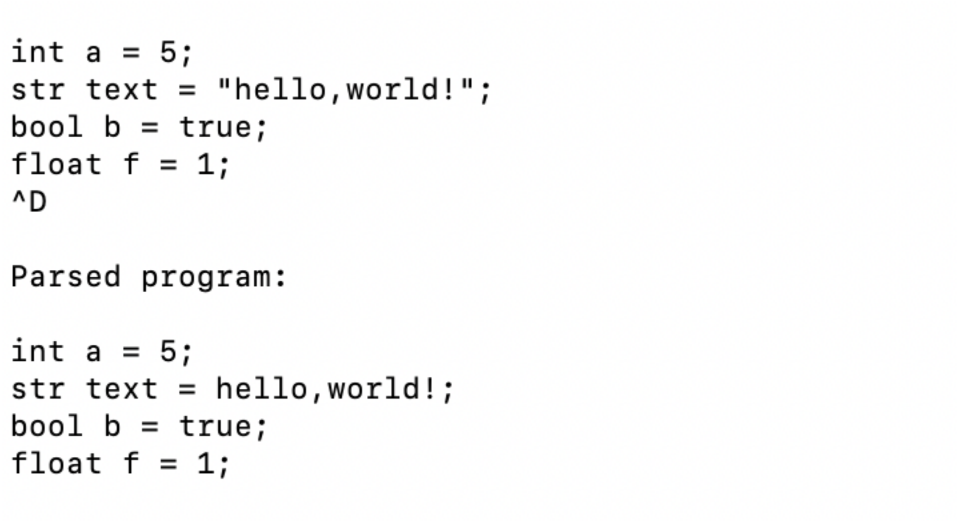
Testing

Primitive Datatypes:

- **Int**
- **Bool**
- **Float**
- **String**



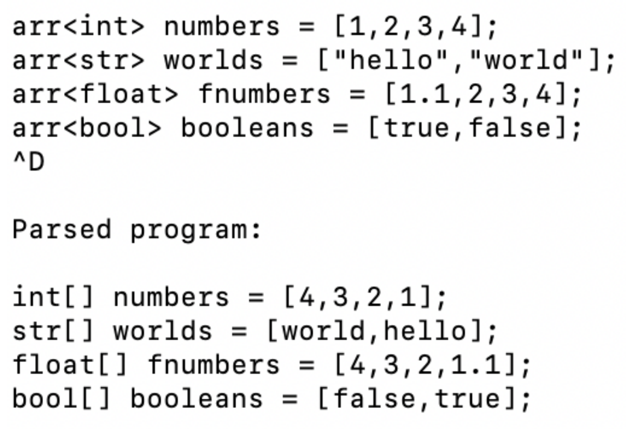
```
int a = 5;
str text = "hello,world!";
bool b = true;
float f = 1;
^D

Parsed program:

int a = 5;
str text = hello,world!;
bool b = true;
float f = 1;
```

Non-primitive datatypes:

- **Array of Int**
- **Array of Bool**
- **Array of Float**
- **Array of String**



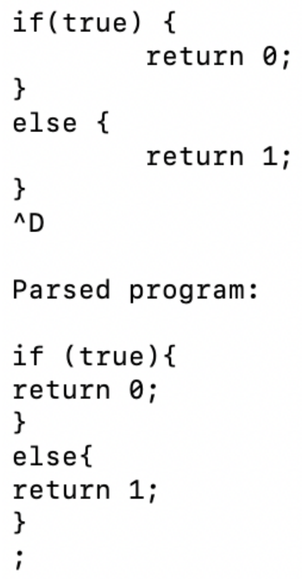
```
arr<int> numbers = [1,2,3,4];
arr<str> worlds = ["hello","world"];
arr<float> fnumbers = [1.1,2,3,4];
arr<bool> booleans = [true,false];
^D

Parsed program:

int[] numbers = [4,3,2,1];
str[] worlds = [world,hello];
float[] fnumbers = [4,3,2,1.1];
bool[] booleans = [false,true];
```

Loops and Conditionals:

- **If**
- **If Else**

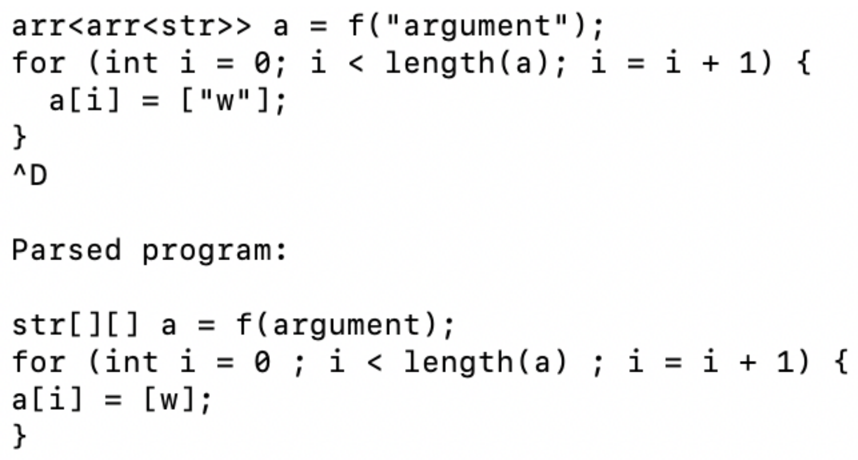


```
if(true) {
        return 0;
}
else {
        return 1;
}
^D

Parsed program:

if (true){
return 0;
}
else{
return 1;
}
;
```

- **For**

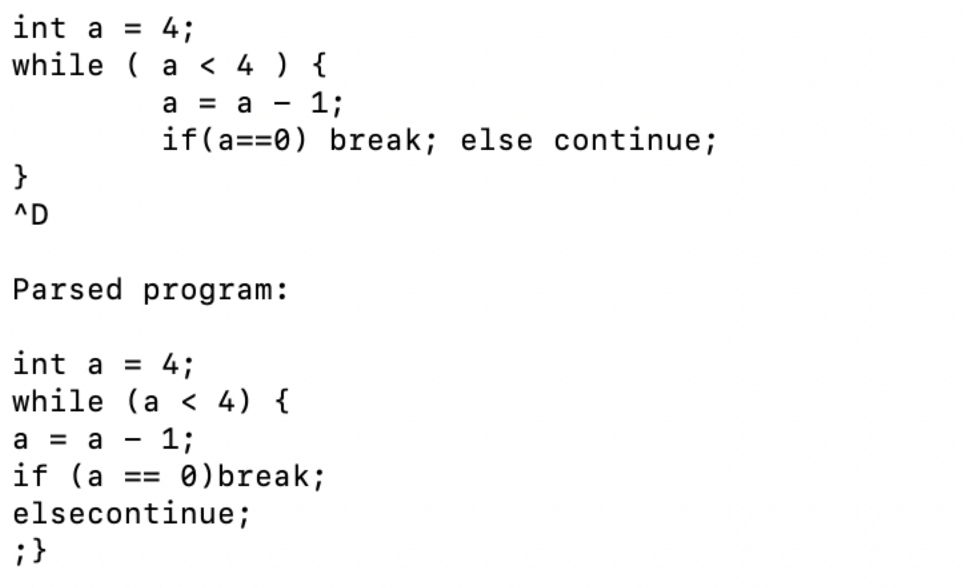


```
arr<arr<str>> a = f("argument");
for (int i = 0; i < length(a); i = i + 1) {
  a[i] = ["w"];
}
^D

Parsed program:

str[][] a = f(argument);
for (int i = 0 ; i < length(a) ; i = i + 1) {
a[i] = [w];
}
```

- **While**
- **Continue**
- **Break**



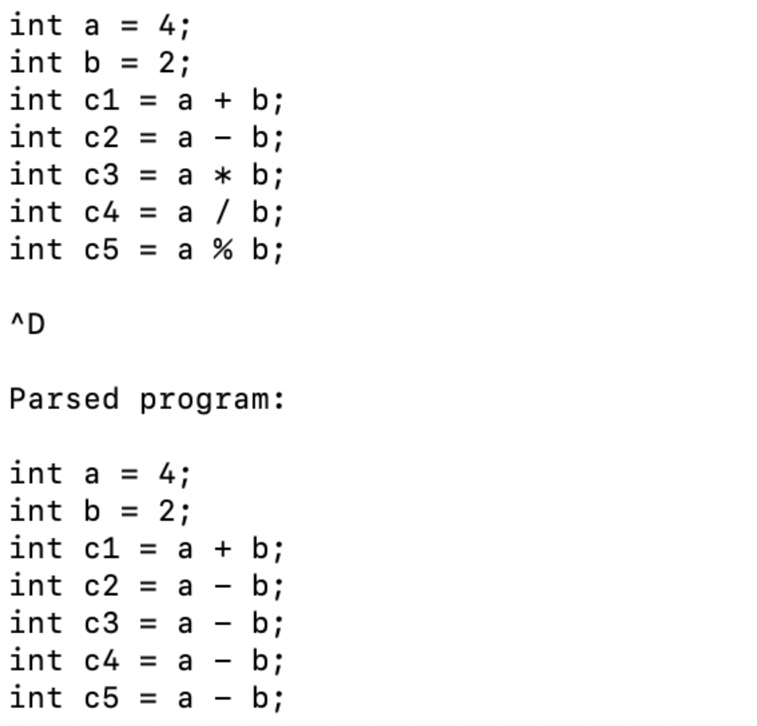
```
int a = 4;
while ( a < 4 ) {
        a = a - 1;
        if(a==0) break; else continue;
}
^D

Parsed program:

int a = 4;
while (a < 4) {
a = a - 1;
if (a == 0)break;
elsecontinue;
;}
```

Binary Operators:

- **Add**
- **Subtract**
- **Multiplication**
- **Division**
- **Modulo**



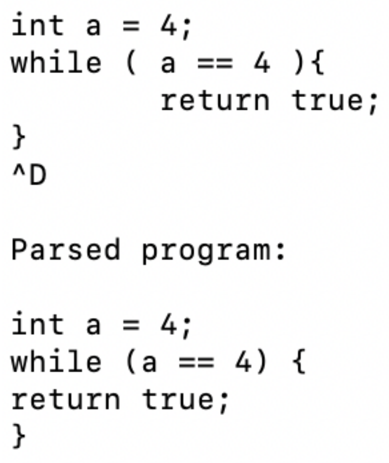
```
int a = 4;
int b = 2;
int c1 = a + b;
int c2 = a − b;
int c3 = a ∗ b;
int c4 = a / b;
int c5 = a % b;

^D

Parsed program:

int a = 4;
int b = 2;
int c1 = a + b;
int c2 = a − b;
int c3 = a − b;
int c4 = a − b;
int c5 = a − b;
```

- **Equal**

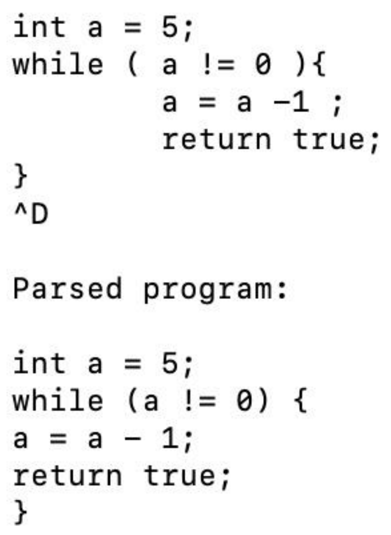


```
int a = 4;
while ( a == 4 ){
        return true;
}
^D

Parsed program:

int a = 4;
while (a == 4) {
return true;
}
```

- **Not Equal**

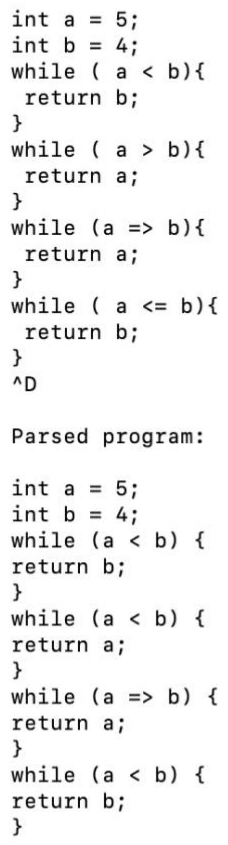


```
int a = 5;
while ( a != 0 ){
        a = a -1 ;
        return true;
}
^D

Parsed program:

int a = 5;
while (a != 0) {
a = a - 1;
return true;
}
```

- **Less**
- **Less than Equals**
- **Greater**
- **Greaer than Equal**

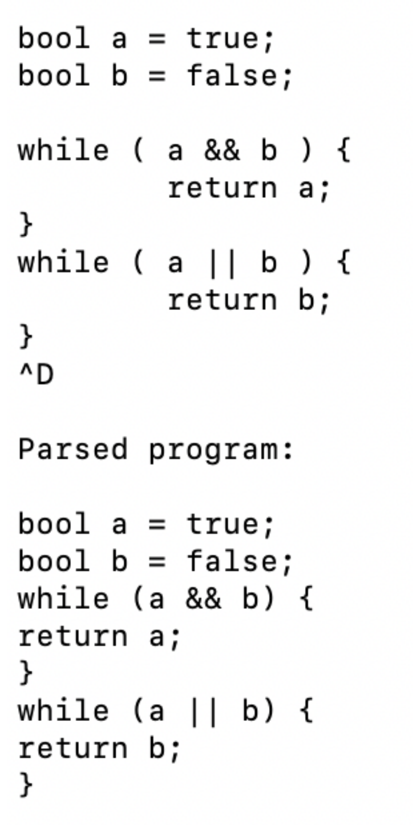


```
int a = 5;
int b = 4;
while ( a < b){
 return b;
}
while ( a > b){
 return a;
}
while (a => b){
 return a;
}
while ( a <= b){
 return b;
}
^D

Parsed program:

int a = 5;
int b = 4;
while (a < b) {
return b;
}
while (a < b) {
return a;
}
while (a => b) {
return a;
}
while (a < b) {
return b;
}
```

- **And**
- **Or**



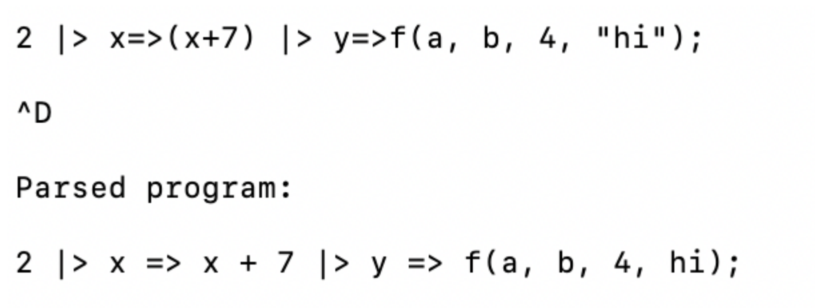
```
bool a = true;
bool b = false;

while ( a && b ) {
        return a;
}
while ( a || b ) {
        return b;
}
^D

Parsed program:

bool a = true;
bool b = false;
while (a && b) {
return a;
}
while (a || b) {
return b;
}
```

- **Pipe**



```
2 |> x=>(x+7) |> y=>f(a, b, 4, "hi");

^D

Parsed program:

2 |> x => x + 7 |> y => f(a, b, 4, hi);
```