CSCI 2132
Software Development

Lecture 20:

Program Organization

Instructor: Vlado Keselj

Faculty of Computer Science

Dalhousie University

Previous Lecture

- Generating permutations (finished)
- Multidimensional arrays as arguments
- **Program Organization:**
- Local variables

Local Static Variables

- Using keyword `static` with local variables
- Have static storage duration, but local scope
- Example:

```
int counter() {
    static int cnt = 0;
    return cnt++;
}
```

- What does the function return if we call it a few times?

External Variables (Global Variables)

- Variables declared outside any function
- Have static storage duration; i.e. stored in *data* part of memory
- Have global scope, i.e., file scope and also visible from other files (using keyword `extern`, to see later)
- If using keyword `static` they will have only file scope

Organizing a C Program in a Single File

```
#include directives
#define directives
type definitions
global variable definitions
function prototypes (except main)
main
other function definitions
```

Example: Decimal to Binary

- Task: Write a program to convert decimal numbers to binary representation
- First attempt:

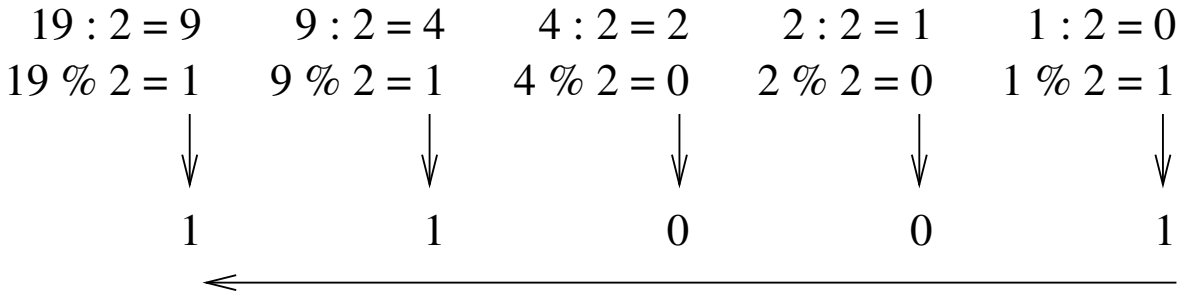
```
While number > 0
    Find out the remainder after
            dividing the number by 2
    Print the remainder
    Divide the number by 2
```

- What is wrong with this pseudo code? Example?

Decimal to Binary: Solution Idea

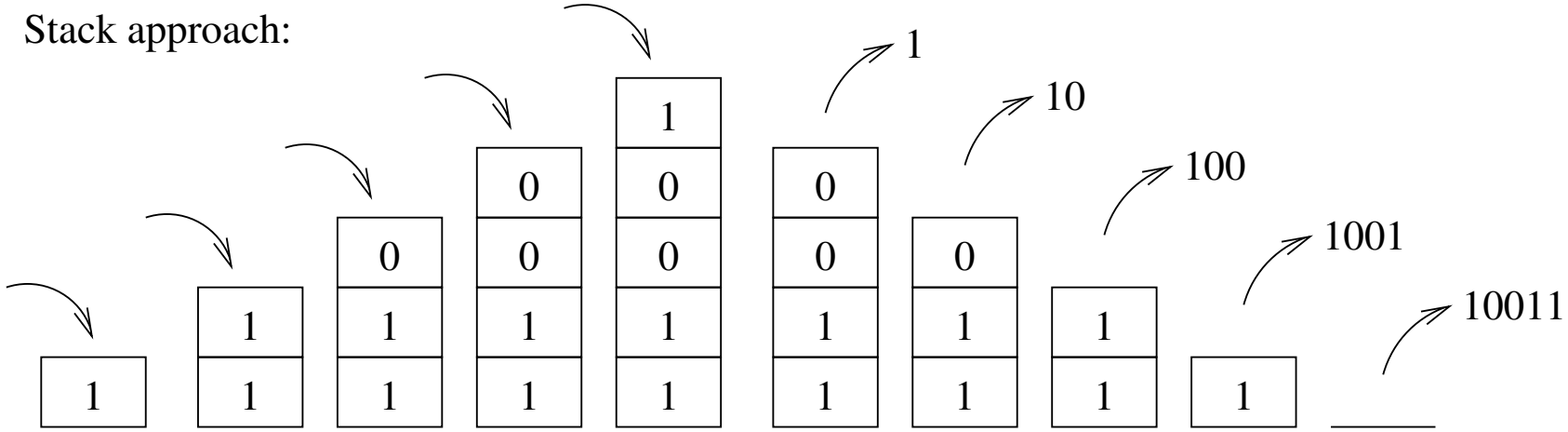
- Use Stack data structure
- Push digits as we go, and later pop and print
- Let us consider an example. . .

Example: Decimal to Binary



19 (decimal) = 10011 (binary)

Stack approach:



Decimal to Binary: Implementation

- Let us take a look at the implementation:

  `~prof2132/public/decimal2binary.c-blanks`

The Fill-in-blanks Code

```
/* Program: decimal2binary.c */
#include <stdio.h>
#include <stdbool.h> /* C99 Standard */
#include <stdlib.h>

#define STACK_SIZE 100

typedef int Bit;

Bit contents[STACK_SIZE];
int top = 0;          /* index to next available spot */

void make_empty();
bool is_empty();
bool is_full();
void push(Bit i);
```

```
Bit pop();
void stack_overflow();
void stack_underflow();

int main() {
  int decimal;
  Bit bit;

  printf("Enter a decimal integer: ");
  scanf("%d", &decimal);

  while (decimal > 0) {
    bit = decimal % 2;
    ___________;
    decimal /= 2;
  }

  printf("This number can be expressed in binary as: ");
```

```
  while (!is_empty())
    printf("%d", ________);

  printf("\n");

  return 0;
}

void make_empty() {
  top = 0;
}

bool is_empty() {
  return top == 0;
}

bool is_full() {
```

```
  return top == STACK_SIZE;
}

void push(Bit i) {
  if (is_full())
    stack_overflow();
  else
    contents[top++] = i;
}

Bit pop(void) {
  if (is_empty())
    stack_underflow();
  else
    return contents[__________];
}

void stack_overflow(void) {
```

```
  printf("Error: stack overflow!\n");
  exit(EXIT_FAILURE);
}

void stack_underflow(void) {
  printf("Error: stack underflow!\n");
  exit(EXIT_FAILURE);
}
```

Avoid Using Global Variables Unnecessarily

- Avoid using global variables unnecessarily
- They make it harder to maintain a program and debug a program
  - Consider changing a variable name throughout a program
- Global variables make it harder to reuse the code
- Consider using two stacks in the previous examples
- How could we approach this?

Blocks and Compound Statements

- A block, or compound statement: `{` *statements* `}`
- Even before C99: could define variables at the beginning of any block, having block scope
- Example

```
if (i < j) {
    int temp = i;
    i = j;
    j = temp;
}
```

- These are local variables with automatic storage duration (on stack)

Scope Example

```
1.  int i;
2.  void f(int i) {
3.    i = 1;
4.
5.    if (i < 0) {
6.      int i;
7.      i = 4;
8.    }
9.    i = 14;
10. }
11.
12. void h() {
13.   i = 5;
14. }
```