

Εργαστήριο Δομές Δεδομένων και Τεχνικές Προγραμματισμού

Εργαστήριο 2 – Μερική απόκρυψη / Makefile

Σκοπός του εργαστηρίου αυτού είναι η εξοικείωση σας με το εργαλείο make και την τεχνική της μερικής απόκρυψης πληροφορίας.

Δίνονται τα αρχεία:

stacktype.h:

```
#ifndef __STACKTYPE_H__
#define __STACKTYPE_H__

typedef int stackelem_t;

#endif
```

test.c

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_ELEMS 10

#include "stacktype.h"

typedef struct {
    int top;
    stackelem_t table[MAX_ELEMS];
} stack_t;

void stack_create(stack_t * stackPtr);
int stack_empty(stack_t stackPtr);
int stack_full(stack_t stack);
stackelem_t stack_pop(stack_t * stackPtr, int *underflow);
void stack_push(stack_t * stackPtr, stackelem_t elem, int *overflow);

/* actually initializes stack; does not create */
void stack_create(stack_t * stackPtr)
{
    stackPtr->top = -1;
}

int stack_empty(stack_t stackPtr)
{
    return (stackPtr.top == -1);
}

int stack_full(stack_t stack)
{
    return (stack.top == MAX_ELEMS - 1);
}

stackelem_t stack_pop(stack_t * stackPtr, int *underflow)
{
    stackelem_t elem;
    if (stack_empty(*stackPtr)) {
        *underflow = 1;
    } else {
        *underflow = 0;
    }
}
```

```

        elem = stackPtr->table[stackPtr->top];
        stackPtr->top--;
    }
    return elem;
}

void stack_push(stack_t * stackPtr, stackelem_t elem, int *overflow)
{
    if (stack_full(*stackPtr)) /* (stackPtr->top == (MAX_ELEMS -1)) */
        *overflow = 1;
    else {
        *overflow = 0;
        stackPtr->top++;
        stackPtr->table[stackPtr->top] = elem;
    }
}

void print_options(void)
{
    printf("\n");
    printf("0. Exit\n");
    printf("1. Create stack\n");
    printf("2. Push integer into stack\n");
    printf("3. Pop integer from stack\n");
    printf("4. Check for empty stack (default)\n");
    printf("Enter your input (0-4): ");
}

int main(void)
{
    char buf[100];
    int option, error, result;
    stack_t stack;
    stackelem_t x;

    do {
        option = -1;
        while (option < 0 || option > 4) {
            print_options();
            fgets(buf, sizeof(buf), stdin);
            sscanf(buf, "%d", &option);
        }
        switch (option) {
            case 1:
                stack_create(&stack);
                printf("\nStack Created!\n");
                break;
            case 2:
                result = 0;
                while (result != 1) {
                    printf("Type an int: ");
                    fgets(buf, sizeof(buf), stdin);
                    result = sscanf(buf, "%d", &x);
                }
                stack_push(&stack, x, &error);
                if (error)
                    printf("\nStack overflow. %d NOT pushed!\n", x);
                break;
            case 3:

```

```

        x = stack_pop(&stack, &error);
        if (error)
            printf("\nEmpty stack!\n");
        else
            printf("\nNumber %d popped successfully!\n", x);
        break;
    case 4:
        if (stack_empty(stack))
            printf("\nStack empty!\n");
        else
            printf("\nStack not empty!\n");
        break;
    }
}
while (option);
return 0;
}

```

Makefile

```

CC=gcc
CFLAGS=-Wall

test: test.o
    $(CC) $(CFLAGS) -o test test.o

test.o: test.c
    $(CC) $(CFLAGS) -c test.c

.PHONY: clean

clean:
    rm -f test test.o

```

Χρήση της εντολής make και συγγραφή Makefile

Η εντολή make είναι ένα εργαλείο που απλοποιεί τη μεταγλώττιση (compilation) και δημιουργία εκτελέσιμων από επί μέρους modules. Το make χρησιμοποιεί κανόνες μεταγλώττισης, που καθορίζονται ως μια λίστα στόχων, από ένα αρχείο με όνομα Makefile. Το make θα ξαναμεταγλωττίσει μόνο αντικείμενα που πρέπει να ξαναμεταγλωττιστούν (object files ή εκτελέσιμα που εξαρτώνται από αρχεία που έχουν μεταβληθεί από την τελευταία φορά που δημιουργήθηκαν τα object files ή τα εκτελέσιμα).

Χρησιμοποιώντας το make

1. Δημιουργήστε ένα Makefile με τους κανόνες που απαιτούνται για τη δημιουργία του εκτελέσιμου. Το αρχείο θα πρέπει να ονομάζεται 'Makefile' ή 'makefile'. Το Makefile θα είναι μοναδικό και θα ενημερώνεται μόνο όταν προστίθενται νέα modules στο πρόγραμμα μας. Τότε θα ενημερώνεται για τους κανόνες και τις εξαρτήσεις των στόχων, και πιθανώς να προστίθενται νέοι στόχοι/κανόνες.
2. Μετά την τροποποίηση των αρχείων πηγαίου κώδικα, ξαναμεταγλωττίστε το εκτελέσιμο πληκτρολογώντας make:

```
$ make
```

Ένας συγκεκριμένος στόχος στο Makefile μπορεί να εκτελεστεί πληκτρολογώντας

```
$ make target_label
```

Για παράδειγμα στο παρακάτω Makefile, για να διαγραφούν τα objects, εκτελέσιμα:

```
$ make clean
```

Δημιουργία Makefile

Ένα Makefile κατά κανόνα ξεκινά με τον ορισμό μεταβλητών που ακολουθείται από ένα σύνολο εγγραφών στόχων για τη δημιουργία συγκεκριμένων στόχων (συνήθως αρχεία .o και εκτελέσιμα αρχεία στη C) ή για την εκτέλεση ενός συνόλου εντολών που σχετίζονται με την ετικέτα στόχου.

Η γενική μορφή ενός στόχου είναι η εξής:

```
# comment
# (note: the <tab> in the command line is necessary for make to work)
target: dependency1 dependency2 ...
    <tab> command
```

Για παράδειγμα:

```
# target entry to build program executable from program and mylib
# object files
#
program: program.o mylib.o
    gcc -o program program.o mylib.o
```

Παράδειγμα Makefile (απλό)

```
# build an executable named myprog from myprog.c
all: myprog.c
    gcc -g -Wall -o myprog myprog.c

clean:
    rm myprog
```

Παράδειγμα με πολλά modules

```
# Define the symbols we might want to change:
CC=gcc
CFLAGS=-g
OBJECTS=processing.o gui.o

my_prog: $(OBJECTS)
    $(CC) $(OBJECTS) -o my_program

processing.o: processing.c
    $(CC) $(CFLAGS) -c processing.c -o processing.o

gui.o: gui.c
    $(CC) $(CFLAGS) -c gui.c -o gui.o
```

Άσκηση 1

Διαχωρίστε τον κώδικα σε διαφορετικά πηγαία αρχεία, ώστε οι συναρτήσεις που αφορούν το χειρισμό της στοίβας να εμπεριέχονται στα αρχεία `stack.c` και `stack.h`. Διαμορφώστε κατάλληλα το `Makefile`.

Άσκηση 2

Τροποποιήστε κατάλληλα το πρόγραμμα ώστε να χειρίζεται αριθμούς με κινητή υποδιαστολή (`float`) αντί για ακεραίους.

Άσκηση 3

Τροποποιήστε κατάλληλα το πρόγραμμα ώστε να χειρίζεται τον εξής τύπο δεδομένων:

```
typedef struct {  
    int a;  
    char c;  
} stackelem_t;
```

Υπόδειξη: Υλοποιήστε κατάλληλες συναρτήσεις `stack_readvalue()` και `stack_printvalue()` για να χειρίζεστε την είσοδο/έξοδο των στοιχείων. Τροποποιήστε τη `main()` ώστε να χρησιμοποιεί αυτές τις νέες συναρτήσεις.