$ cat head

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Лабораторная работа №25-26 \*

\* \*

\* Утилита make. \*

\* Абстрактные типы данных \*

\* \*

\* Работу выполнил: \*

\* Стеснягин Семён \*

\* Группа: 80-102Б \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$ cat lab26.cpp

#include <time.h>

#include "udt.h"

int main() {

Stack S1, S2;

srand(time(NULL));

Init(S1);

int v;

for (int i = 0; i < 10; i++) {

v = rand()%10;

Push(S1, v);

}

Display(S1);

Init(S2);

for (int i = 0; i < 7; i++) {

v = rand()%11;

Push(S2, v);

}

Display(S2);

Append(S1, S2);

Display(S1);

bubble\_sort(S1);

Display(S1);

}

$ cat udt.h

#include<stdio.h>

#include<stdlib.h>

#define N 100

struct Stack {

int first;

int body[N];

};

void Init(Stack &S);

int Empty(Stack S);

void Push(Stack &S, int V);

int Pop(Stack &S);

int Top(Stack S);

int Size(Stack S);

void Display(Stack S);

void Cat(Stack &S1, Stack &S2);

void Append(Stack &S1, Stack &S2);

void bubble\_sort(Stack &S);

$ cat udt.cpp

#include "udt.h"

void Init(Stack &S) { S.first = 0; }

int Empty(Stack S) { return S.first == 0; }

void Push(Stack &S, int V) {

if (S.first == N) printf("STACK IS OVERFLOW");

else S.body[S.first++] = V;

}

int Pop(Stack &S) {

if (Empty(S)) printf("STACK IS EMPTY");

else return S.body[--S.first];

}

int Top(Stack S) {

if (Empty(S)) printf("STACK IS EMPTY");

else return S.body[S.first - 1];

}

int Size(Stack S) { return S.first; }

void Display(Stack S) {

printf("[ ");

for (int i = 0; i < S.first; i++) printf("%d ", S.body[i]);

printf("]\n");

}

void Cat(Stack &S1, Stack &S2) {

int V;

if (!Empty(S2)) {

V = Pop(S2);

Cat(S1, S2);

Push(S1, V);

}

}

void Append(Stack &S1, Stack &S2) {

int V;

if (Empty(S1)) Cat(S1, S2);

else {

V = Pop(S1);

Append(S1, S2);

Push(S1, V);

}

}

void bubble\_sort(Stack &S)

{

for (int i = 0; i < S.first-1; i++) {

bool swapped = false;

for (int j = 0; j < S.first-i-1; j++) {

if (S.body[j] > S.body[j+1]) {

int b = S.body[j];

S.body[j] = S.body[j+1];

S.body[j+1] = b;

swapped = true;

}

}

if(!swapped)

break;

}

}

$ cat makefile

CC = gcc

CFLAGS = -std=c++11 -pedantic -Wall -g

OBJ = lab26.o udt.o

PROG = program

build: lab26.o udt.o

$(CC) $(CFLAGS) -o $(PROG) $(OBJ)

lab26.o: lab26.cpp

$(CC) $(CFLAGS) -c lab26.cpp

udt.o: udt.cpp

$(CC) $(CFLAGS) -c udt.cpp

clean:

rm $(PROG) $(OBJ)

$ make build

gcc -std=c++11 -pedantic -Wall -g -c lab26.cpp

gcc -std=c++11 -pedantic -Wall -g -c udt.cpp

udt.cpp: In function 'int Pop(Stack&)':

udt.cpp:15:1: warning: control reaches end of non-void function [-Wreturn-type]

}

^

udt.cpp: In function 'int Top(Stack)':

udt.cpp:20:1: warning: control reaches end of non-void function [-Wreturn-type]

}

^

gcc -std=c++11 -pedantic -Wall -g -o program lab26.o udt.o

$ ./program

[ 0 3 2 0 9 0 2 0 9 8 ]

[ 3 1 7 9 1 9 2 ]

[ 3 1 7 9 1 9 2 0 3 2 0 9 0 2 0 9 8 ]

[ 0 0 0 0 1 1 2 2 2 3 3 7 8 9 9 9 9 ]

$ ./program

[ 4 1 4 9 2 7 0 9 1 2 ]

[ 3 1 3 8 6 0 2 ]

[ 3 1 3 8 6 0 2 4 1 4 9 2 7 0 9 1 2 ]

[ 0 0 1 1 1 2 2 2 3 3 4 4 6 7 8 9 9 ]

$ make clean

rm program lab26.o udt.o

$ ls

Desktop lab21\_var14.sh makefile

New Empty File lab22.tex tree.h

a.out lab23\_var14.c udt.cpp

head lab24\_var14.c udt.h

kp6\_var34 lab24\_var1