Технически университет – София

**Курсова работа**

по ПИК2

Задание №: 10

**Изготвил:** Цветелина Георгиева

Специалност: КСИ

Група: 41

Фак. №: 121220040

**Преподавател:** Момчил Петков

2. Задание:

Да се разработи програма с меню тип пощенска кутия:

1. Писмата на даден потребител се съхраняват в двоичен файл. Всяко писмо съдържа следните данни:

• Уникален идентификационен номер

• Получател

• Копие до

• Подател

• Заглавие

• Съдържание

• Дата на получаване - записана във формата ДД.ММ.ГГГГ HH:MM:SS. 2. Програмата трябва да предоставя на потребителя следната функционалност:

а) добавяне нов мейл към списъка;

б) показване на опростен списък с мейлите в пощенската кутия: номер, подател, заглавие, дата на получаване.

в) показване на съдържанието на мейл по даден идентификационен номер;

г) Изчиства всички спам писма. За спам писмо се приема писмо,

в което съотношението на забранените думи към общия брой думи е повече от 1/5-та.

Забранените думи са дефинирани в текстовия файл: „bad\_words.txt”;

3. Данните да се поддържат в динамична структура - едносвързан списък в оперативната памет на ПК.

3. Описание на потребителски дефинираните функции:

List \*addNewEmail() – Функцията дава възможност на потребителя да добави нов имейл към вече съществуващите във файла Gmail.bin. В нея той въвежда уникален идентификационен номер, получател, до кого да изпратят копие, кой е подателя, заглавието на писмото, неговото съдържание и датата.

int BadWord(char \*word) – Функцията проверява за налични спам думи. Тя получава като параметър дума от съдържанието на един имейл във файла Gmail.bin и я сравнява с всички думи написани в bad\_words.txt. При наличие на съвпадение връща 1, в противен случай 0.

List \*delete\_spam(List \*Spam) – Функцията служи за триене на спам имейли. За параметри получава указател сочещ към намерения спам имейл. Обхожда листът до намирането на спам, при намерен такъв го трие и връща указател към началото на променения списък.

4. Общо описание на функционирането на програмата:

Програмата работи като меню с 5 опции. Първата дава възможност на потребителя да добави нов имейл към вече съществуващите, като за тази цел е необходимо коректно да въведе необходимите данни за един имейл. С дадената от него информация се запълва структура, което се добавя към списъка с имейл. Втората опция показва на потребителя опростен списък с всички имейли във файла Gmail.bin, ако има такива, ако няма връща съобщение за грешка. Чрез третата възможност можем да видим съдържанието на имейл, чрез неговия уникален идентификационен номер. Четвъртата обхожда имейлите и проверява дали сред тях има спам такива и ако има ги изтрива, ако няма връща съобщение, че няма такива. С петата опция излизаме от програмата. Осъществена е посредством while цикъл, който ще върти докато не натиснем 5 или не ни излезе съобщение за грешка.

5 .Код на програмата:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

struct email{

int id;

char poluchatel[50];

char copy\_to[50];

char podatel[50];

char title[100];

char content[500];

int day;

int month;

int year;

int hour;

int min;

int sec;

};

typedef struct email email;

struct List

{

email email;

struct List\* next;

};

typedef struct List List;

List \*addNewEmail()

{

List \*new\_email = NULL;

new\_email = (List\*)malloc(sizeof(List));

if(new\_email==NULL)

{

perror("Not enough memory for list.");

exit(1);

}

printf("Enter your Email: \n");

printf("Enter id of email: ");

scanf("%d", &new\_email->email.id);

printf("Enter poluchatel of email: ");

fflush(stdin);

if(fgets(new\_email->email.poluchatel,50,stdin)!=NULL){

new\_email->email.poluchatel[strlen(new\_email->email.poluchatel)-1]='\0';

}

printf("Enter copy\_to of email: ");

fflush(stdin);

if(fgets(new\_email->email.copy\_to,50,stdin)!=NULL){

new\_email->email.copy\_to[strlen(new\_email->email.copy\_to)-1]='\0';

}

printf("Enter podatel of email: ");

fflush(stdin);

if(fgets(new\_email->email.podatel,50,stdin)!=NULL){

new\_email->email.podatel[strlen(new\_email->email.podatel)-1]='\0';

}

printf("Enter title of email: ");

fflush(stdin);

if(fgets(new\_email->email.title,100,stdin)!=NULL){

new\_email->email.title[strlen(new\_email->email.title)-1]='\0';

}

printf("Enter content of email: ");

fflush(stdin);

if(fgets(new\_email->email.content,500,stdin)!=NULL){

new\_email->email.content[strlen(new\_email->email.content)-1]='\0';

}

printf("Enter date of email:\n ");

printf("FORMAT: DD.MM.YYYY HH:MM:SS. ");

scanf("%d", &new\_email->email.day);

printf(".");

scanf("%d" ,&new\_email->email.month);

printf(".");

scanf("%d" ,&new\_email->email.year);

printf("\t");

scanf("%d" ,&new\_email->email.hour);

printf(":");

scanf("%d" ,&new\_email->email.min);

printf(":");

scanf("%d" ,&new\_email->email.sec);

new\_email->next = NULL;

return new\_email;

}

int BadWord(char \*word)

{

FILE\* fp;

if((fp = fopen("bad\_words.txt", "r"))==NULL)

{

perror("Problem with bad\_words.txt.\n");

exit(5);

}

char bad\_words\_from\_file[50];

while(fscanf(fp,"%s",bad\_words\_from\_file) != EOF)

{

if(strcmp(word,bad\_words\_from\_file) == 0)

{

fclose(fp);

return 1;

}

}

fclose(fp);

return 0;

}

List \*delete\_spam(List \*Spam)

{

FILE \*fin;

if((fin=fopen("Gmail.bin","rb")) == NULL)

{

perror("Cannot open a bin file for read.");

exit(1);

}

List \*root = NULL;

root = (List\*)malloc(sizeof(List));

if(root == NULL)

{

perror("Memory error (root).");

exit(4);

}

while(fread(&root->email,sizeof(email),1,fin)==1){}

if(root == NULL)

{

printf("No emails in file.\n");

return root;

}

List\* temp = root;

List\* prev = root;

for(; temp != NULL; prev=temp, temp=temp->next)

{

if(temp == Spam)

{

if(root == Spam){

root = root->next;

free(Spam);

fclose(fin);

return root;

}

prev->next = temp->next;

free(temp);

temp = prev;

fclose(fin);

return temp;

}

}

fclose(fin);

return root;

}

int main()

{

int options = 0;

while(options!=5)

{

printf(" 1 -> If ypu have to add a email.\n 2 -> If you have to visit all emails\n 3 -> if you should s visit a email.\n 4 -> Delete all spam emails.\n 5-> End of the program.\n");

scanf("%d", &options);

if(options == 1){

FILE \*fp;

if((fp=fopen("Gmail.bin","ab+"))==NULL){

perror("Cannot open a bin file for add.");

exit(2);

}

List \*root=NULL;

root=addNewEmail();

if(fwrite(root,sizeof(email),1,fp)!=1){

perror("Write error");

exit(3);

}

fclose(fp);

} else if(options == 2)

{

FILE \*fin;

if((fin=fopen("Gmail.bin","rb"))==NULL){

perror("Cannot open a bin file for read.");

exit(3);

}

List \*all\_emails = NULL;

all\_emails = (List\*)malloc(sizeof(List));

if(all\_emails == NULL)

{

perror("Memory error (all\_emails).");

exit(4);

}

while(fread(&all\_emails->email,sizeof(email),1,fin)==1)

{

printf("ID: %d\nPodatel: %s\nTitle: %s\nDate: %02d.%02d.%d\t%02d:%02d:%02d.\n\n", all\_emails->email.id, all\_emails->email.podatel, all\_emails->email.title, all\_emails->email.day, all\_emails->email.month, all\_emails->email.year, all\_emails->email.hour, all\_emails->email.min, all\_emails->email.sec);

}

fclose(fin);

}else if(options == 3)

{

int my\_id;

int search = 0;

printf("ID you are looking for: ");

scanf("%d", &my\_id);

FILE \*fs;

if((fs=fopen("Gmail.bin","rb"))==NULL){

perror("Cannot open a bin file for search.");

exit(5);

}

List \*search\_email = NULL;

search\_email = (List\*)malloc(sizeof(List));

while(fread(&search\_email->email,sizeof(email),1,fs) == 1){

if(my\_id == search\_email->email.id)

{

printf("\nContent: %s\n", search\_email->email.content);

search++;

}

}

if(search==0)

{

printf("\nNo email with this id.\n");

}

fclose(fs);

}else if(options == 4)

{

FILE \*fsp;

if((fsp=fopen("Gmail.bin","rb"))==NULL)

{

perror("Cannot open a bin file for read.");

exit(3);

}

List \*spam\_emails = NULL;

spam\_emails = (List\*)malloc(sizeof(List));

if(spam\_emails == NULL)

{

perror("Memory error (spam\_emails).");

exit(4);

}

while(fread(&spam\_emails->email,sizeof(email),1,fsp)==1)

{

int all\_words\_count = 0;

int bad\_words\_count = 0;

int test;

char \*word = strtok(spam\_emails->email.content," ");

while(word != NULL)

{

test = BadWord(word);

if(test == 1)

{

all\_words\_count++;

bad\_words\_count++;

}else if(test == 0)

{

all\_words\_count++;

}

word = strtok(NULL," ");

}

float result = (float)bad\_words\_count/all\_words\_count;

if(result > 1/5)

{

delete\_spam(spam\_emails);

printf("Spam email are deleted.\n");

}else

{

printf("mail is not spam.\n");

}

}

fclose(fsp);

}

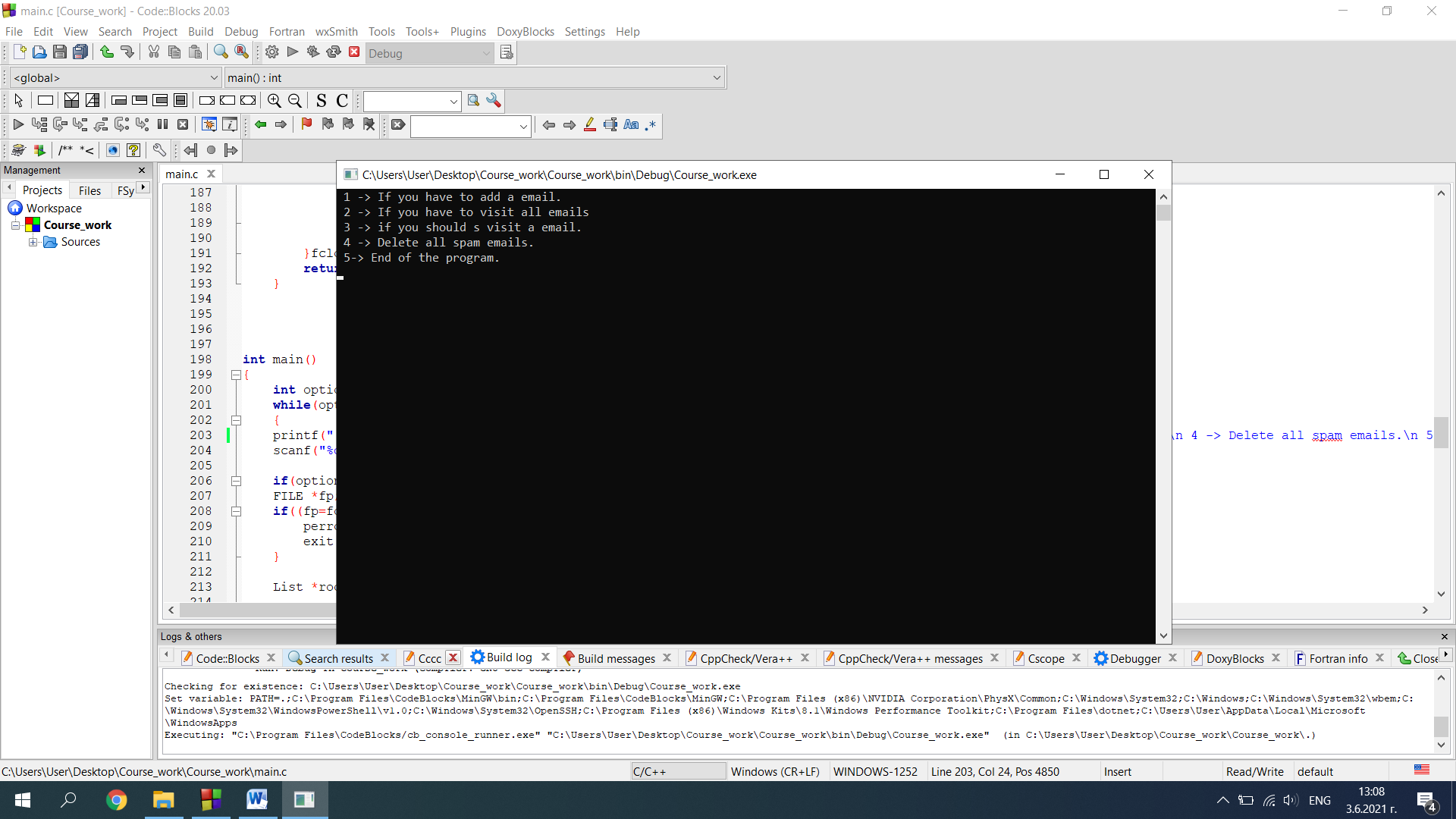
}

return 0;

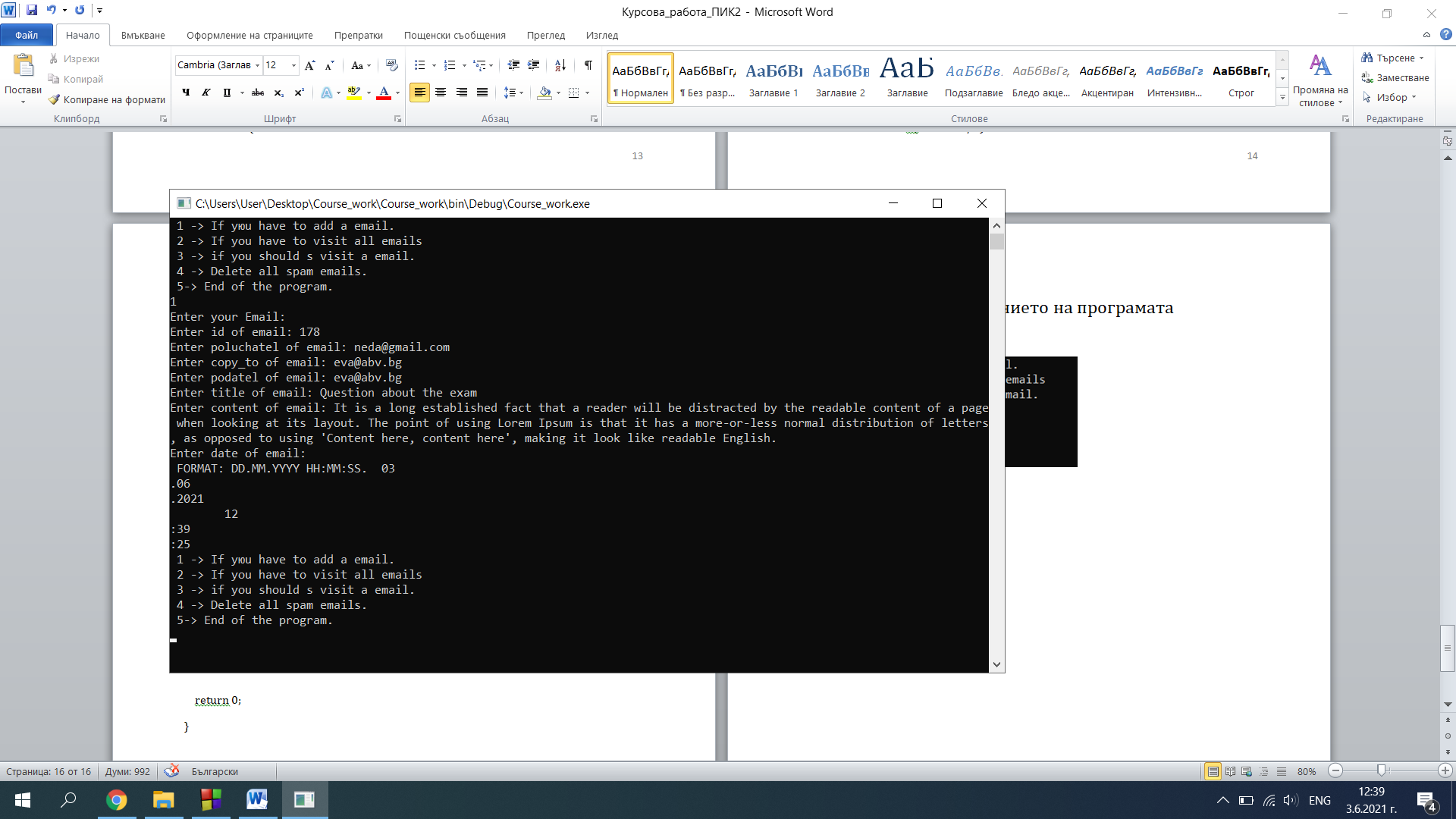
}

6. Резултати от изпълнението на програмата

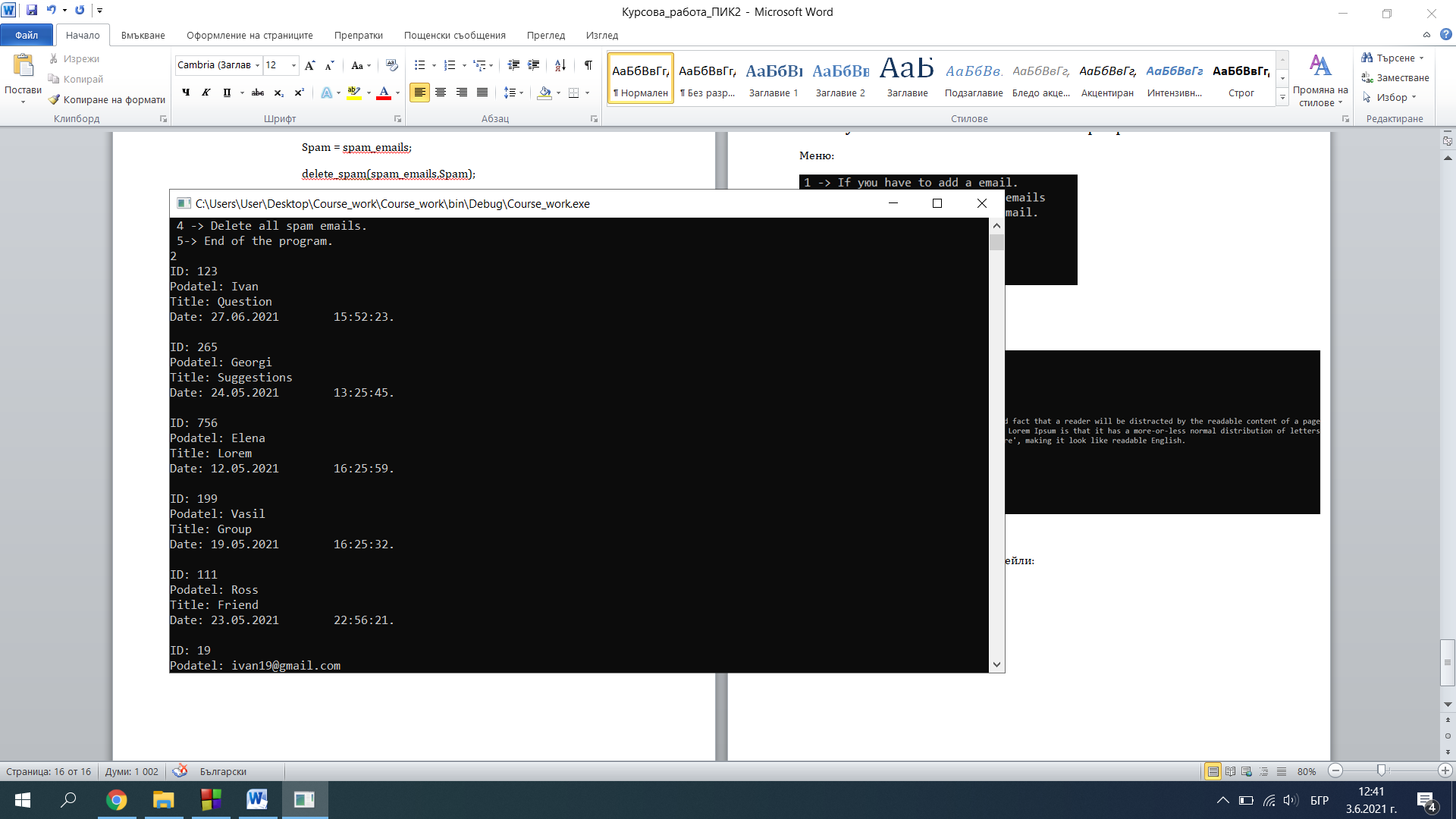
Меню:



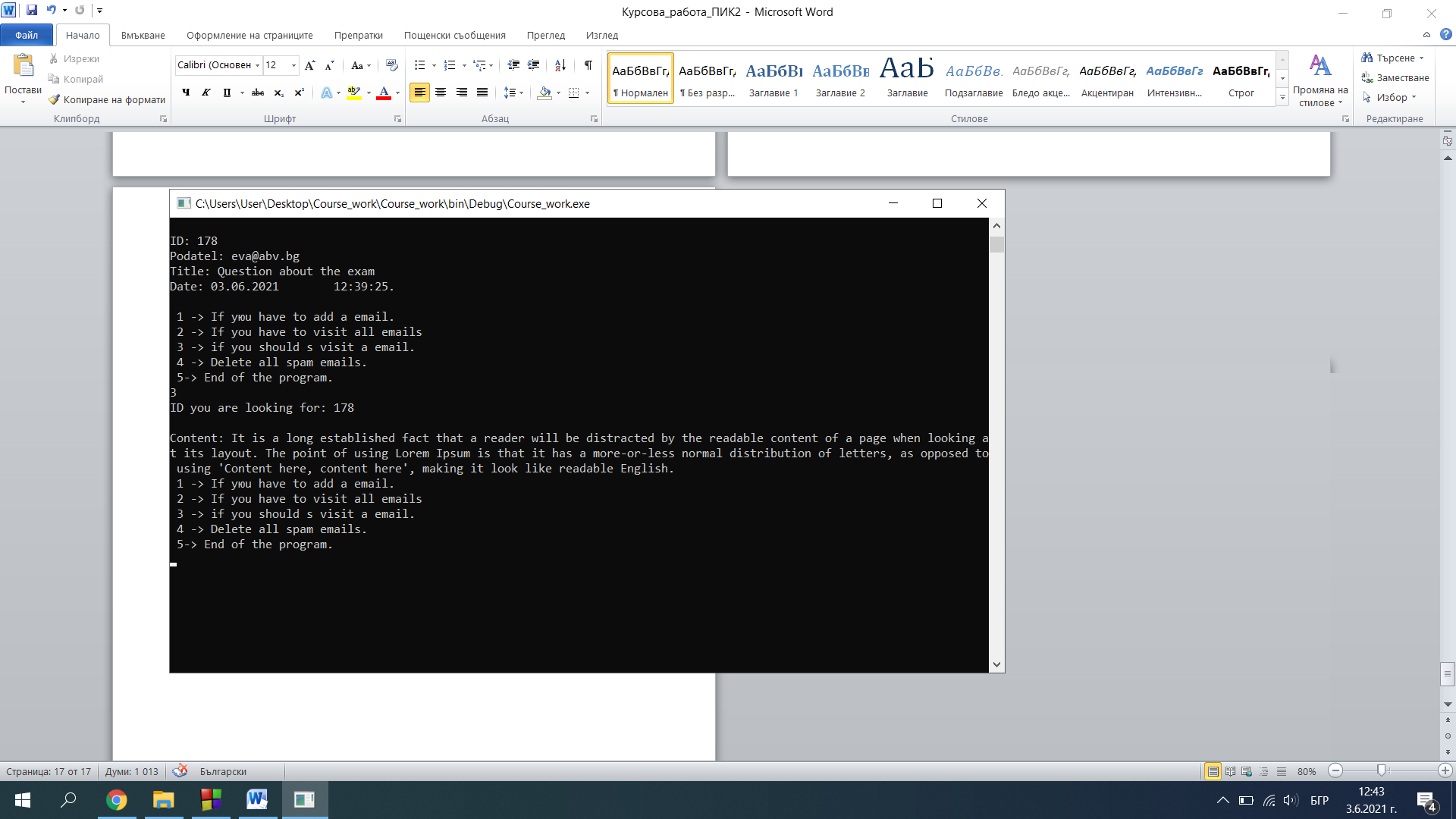
1 -> Добавяне на имейл:



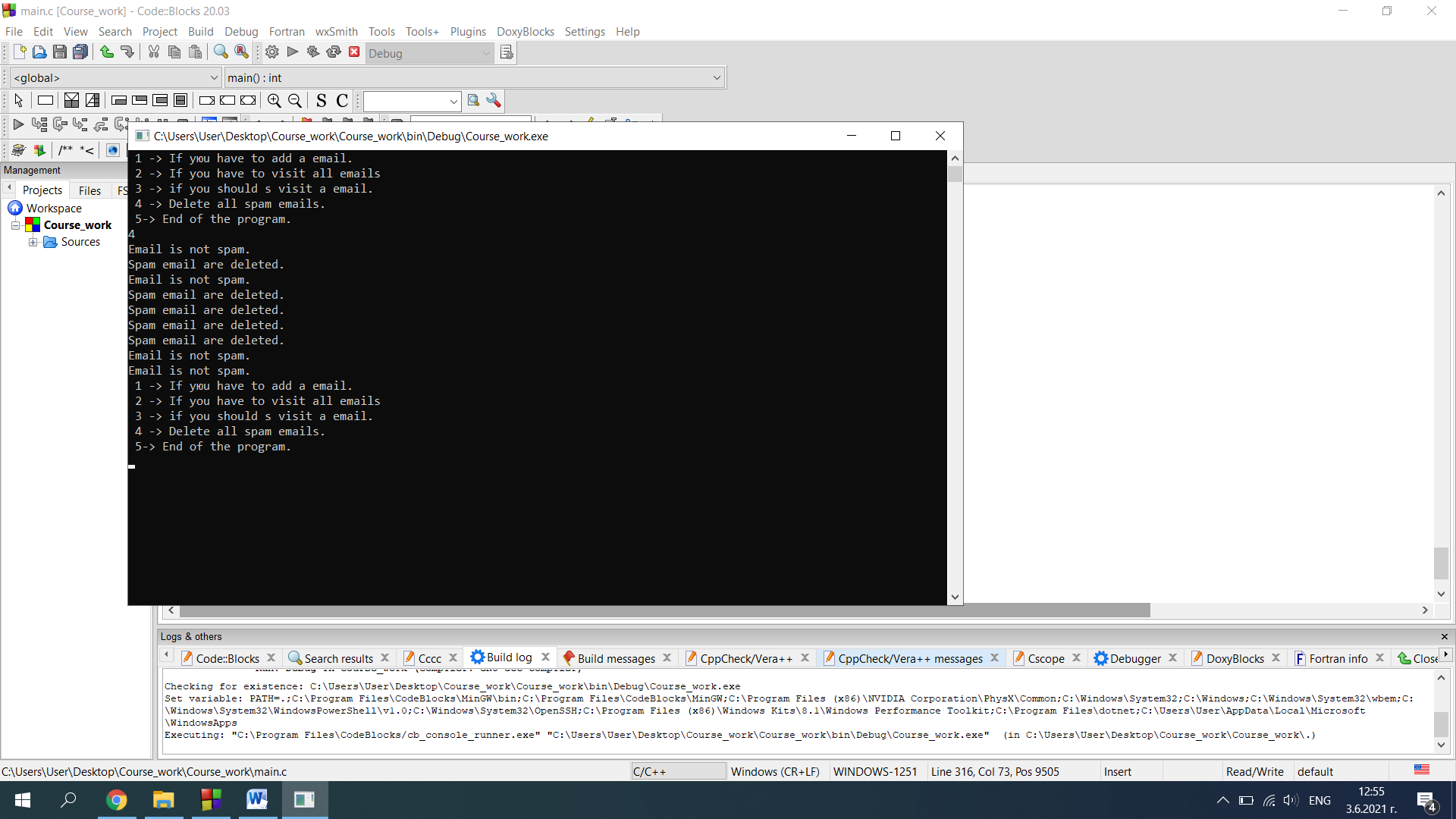
2 -> Преглед на опростен списък с имейли:



3-> Показване на съдържанието на имейла по въведен уникален идентификационен номер



4 -> Проверява дали има въведени спам писма и при откриване на такива ги изтрива



5-> Изход от програмата

