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

typedef struct _Conta{ int num; char nome[20]; float saldo; } Conta;

typedef struct Data { int dia, mes, ano; } Data;

typedef struct _Aluno { int nroUSP; char nome[40], curso[20]; Data
nascimento; } Aluno;

void imprime_vetor_stack(int* vet, int n) {
	for (int p=0; p < n; p++) printf("[%d] %d (%p)\n", p, *(vet+p),
vet+p);
	printf("\n");
}

void imprime_matriz_stack(int* mat, int x, int y) {
	for (int i = 0; i < x; i++)
		for (int j = 0; j < y; j++)
			printf("[%d,%d] %d (%p)\n", i, j, *(mat + i * y
+ j), (mat + i * y + j)); //valor (endereco)
}

void atribuiMatriz(int** m, int row, int col) {
	for (int i = 0; i < row; i++)
		for (int j = 0; j < col; j++)
			m[i][j] = (i+1)*(j+2);
}

void imprimeMatriz(int** m, int row, int col) {
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < col; j++)
			printf("%d ", m[i][j]);
		printf("\n");
	}
}

int* alocaVetorA(int N) {
	return (int*) malloc(N * sizeof(int));
}

void alocaVetorB(int** p, int N) { *p = (int*) malloc(N * sizeof(int)); }
char* alocaString(int N) { return (char*) malloc(N * sizeof(char)); }
void desalocaVetor(int* vet) { free(vet); }
void desalocaString(char* str) { free(str); }

int** alocaMatriz(int row, int col) {
	int** m = NULL;
	m = (int**) calloc(row, sizeof(int *)); // aloca N linhas como
vetores de int*
	for (int i = 0; i < row; i++)
		m[i] = (int *) calloc(col, sizeof(int)); // para cada linha,
M colunas de tipo int
	return m;
}

void desalocaMatriz(int row, int col, int** m) {
	for (int i = 0; i < row; i++) // cada vetor (coluna) dentro da matriz
tem que ser liberado na heap
		free(m[i]);
	free(m); // libera o vetor de linhas na heap
}

Aluno newAluno(int nusp, char nome[], char curso[], int dia, int mes, int
ano) {
	Aluno b;

	b.nroUSP = nusp;
	strcpy(b.nome, nome);
	strcpy(b.curso, curso);
	b.nascimento.dia = dia;
	b.nascimento.mes = mes;
	b.nascimento.ano = ano;

	return b;
}

void newAluno2(Aluno* alu, int nusp, char nome[], char curso[], int dia,
int mes, int ano) {
	alu->nroUSP = nusp;
	strcpy(alu->nome, nome);
	strcpy(alu->curso, curso);
	alu->nascimento.dia = dia;
	alu->nascimento.mes = mes;
	alu->nascimento.ano = ano;

	return;
}

Conta* criaVetorContas(int N) {
	return (Conta*) malloc(sizeof(Conta) * N);
}

void AulaStack() {
	int vet[4] = {1, 2, 3, 4};
	int mat[3][2] = { {10, 11}, {12, 13}, {14, 15}};

	printf("\n--Vetor--\n");
	imprime_vetor_stack(vet, 4);

	printf("\n--Matriz--\n");
	imprime_matriz_stack(*mat, 3, 2);

	printf("\n--Outra Matriz--\n");
	int i, j;
	for (i = 0; i < 3; i++) {
		for (j = 0; j < 2; j++) {
			printf("[%d,%d] %d (%p)\n", i, j, *(*(mat+i)
+j), *(mat+i)+j);
		}
	}

	printf("\nOutras formas do vetor:\n");
	printf("1. %d\n", vet[1]);
	printf("2. %d\n", *(vet+1));
	printf("3. %d\n", *vet+1);
	printf("4. %p\n", vet);
	printf("5. %p\n", vet+1);

	printf("\nOutras formas da matriz:\n");
	printf("1. %d\n", mat[0][1]);
	printf("2. %d\n", *(*(mat+0)+1));
	printf("3. %p\n", *mat);
	printf("4. %p\n", *(mat+1));
	printf("5. %p\n", mat);
	printf("6. %p\n", mat+1);
}

void AulaHeap() {
	int* ip; // stack (alocacao automatica)
	ip = (int*) malloc(1 * sizeof(int)); // alocacao dinamica (heap)
```

```
	*ip = 93;

	printf("ip\t%p\t%p\t%d\n", &ip, ip, *ip);
	free(ip);

	int* vetor = calloc(50, sizeof(int)); //aloca e manda 0
	free(vetor);
}

void AulaString() {
	char* string1 = alocaString(500);

	string1[0] = 'q'; string1[1] = 'u';
	string1[2] = 'a'; string1[3] = 'r';
	string1[4] = 't'; string1[5] = 'u';
	string1[6] = 's'; string1[7] = ' ';
	string1[8] = '\0';
	printf("%s\n", string1);
	strcpy(string1, "quartus");
	printf("%s\n", string1);
	gets(string1);
	printf("%s\n", string1);
	free(string1);
}

int numContas;
void CriaNovaConta(Conta** vet) {
	numContas++;
	*vet = realloc((*vet), sizeof(Conta) * numContas);

	(*vet)[numContas-1].num   = numContas;
	(*vet)[numContas-1].saldo = 0.0;

	printf("Digite o nome: ");
	fgets((*vet)[numContas-1].nome, 20, stdin);
	int i = -1;
	char c;

	while((*vet)[numContas-1].nome[++i] != '\0');
	(*vet)[numContas-1].nome[i] = '\0';
}


void ListaContas(Conta* c) {
	printf("\n\n");
	printf("=======\n");
	printf(" Lista\n");
	printf("=======\n");
	for (int i = 0; i < numContas; i++) {
		printf("\nConta %d\n", i+1);
		printf("Numero: %d\n", c[i].num);
		printf("Nome: %s\n", c[i].nome);
		printf("Saldo: %.2f\n", c[i].saldo);
	}
}

void ConsultaSaldo(Conta* vet) {
	int con;
	printf("\nDigite o saldo que deseja consultar: ");
	scanf("%d", &con);

	int i;
	for (i = 0; i < numContas; i++) {
		if (vet[i].num == con) {
			printf("%d");
		}
	}
}

void AulaStruct() {
	//Cada linha : registro
	//Cada coluna: campo
	Aluno a = newAluno(4482145, "Jorginho", "Agronomia", 10, 10,
2010);
	Aluno b;
	newAluno2(&b, 4482145, "Jorginho", "Agronomia", 10, 10,
2010);
	Aluno c = a; //copia todos os dados, porque nao tem nenhum
int* ou algo do tipo
	Conta* vet = NULL;

	int opc = 0;
	do {
		scanf("%d", &opc); getchar();
		switch(opc) {
			case 1:
				CriaNovaConta(&vet);
				break;
			case 2:
				ListaContas(vet);
				break;
		}
	} while (opc != 4);

	free(vet);
	//criaVetorContas(100);

}

void AulaArquivo() {
	FILE* fpr, * fps;
	char* ign[101], msg[101];
	int qtLida = 0, qtIgn = 0, qt = 0;
	for (qt = 0; qt < 100; qt++) ign[qt] = malloc(51 * sizeof(char));

	char arq[101]; scanf(" %s", arq);
	fpr = fopen(arq, "r");
	fps = fopen("saida.txt", "w");

	int i = 0;
	while(scanf(" %s", ign[i++]) != EOF);

	int ind, existe = 0;
	while(fscanf(fpr, " %s", msg) != EOF) {
		existe = 0;
		for (ind = 0; ind < i; ind++)
			if (strcmp(msg, ign[ind]) == 0) { existe = 1;
break; }

		if (existe) qtIgn++;
		else fprintf(fps, "%s ", msg);
		qtLida++;
	}
	printf("Palavras lidas: %d\nPalavras ignoradas: %d\n", qtLida,
qtIgn);
	fclose(fpr); fclose(fps);
}
```