Języki i paradygmaty programowania 1 – studia stacjonarne 2024/25

Lab 10. Struktury danych: kolejka (bufor FIFO), stos (bufor LIFO), lista jednokierunkowa, lista dwukierunkowa. Unie.

1. Zapoznaj się ze strukturami danych: kolejka (FIFO), stos (LIFO), lista jednokierunkowa, lista dwukierunkowa.

2. Zaimplementuj listę jedno- i dwukierunkową (w osobnych projektach) bazując na strukturach języka C (`struct`).
   W pliku tekstowym zapisane są następujące dane o wielu studentach:
   `imię, nazwisko, rok urodzenia, adres zamieszkania, kwota stypendium.`

   Przykładowa struktura dla listy jednokierunkowej i dwukierunkowej:

```
typedef struct stud {
       char* imie;
       char* nazwisko;
       int rok;
       char* adres;
       double stypendium;
       struct stud* n;
       }STUDENT;
```

```
typedef struct stud {
       char* imie;
       char* nazwisko;
       int rok;
       char* adres;
       double stypendium;
       struct stud* p;
       struct stud* n;
       }STUDENT;
```

   Zapisz te dane do listy struktur i napisz funkcje pozwalające na:
   a) Inicjalizację listy – odczyt danych z pliku;
      `void list1(FILE* plik);`
   b) Wyświetlania zawartości listy;
      `void dispList1();`
   c) Wybranie spośród wszystkich osób tego studenta który pobiera największe stypendium;
      `STUDENT* max_s();`
   d) Dodania elementu na początek listy;
      `void addHeadList1();`
   e) Dodania elementu na koniec listy;
      `void addTailList1();`
   f) Zliczania ilości elementów listy;
      `int lenList1();`
   g) Usunięcia elementu z początku listy;
      `void remHeadList1();`
   h) Usunięcia elementu z końca listy.
      `void remTailList1();`

   Implementacja niektórych funkcjonalności (np. h) będzie uciążliwa szczególnie w przypadku listy jednokierunkowej - proszę wspomagać się dodatkowymi wskaźnikami tymczasowymi i przeszukiwaniem listy od początku.

Fragment programu - przykładowa funkcja tworząca listę jednokierunkową:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#pragma warning(disable: 4996 6387)
#define MAX_BUFOR 80

typedef struct stud {
	char* imie;
	char* nazwisko;
	int rok;
	char* adres;
	double stypendium;
	struct stud* n;
}STUDENT;

STUDENT* H = NULL;
FILE* fd = NULL;

void list1(FILE* plik){
	unsigned len;
	STUDENT* B = NULL, * P = NULL;
	char bufor[MAX_BUFOR];
	//H = (STUDENT*)NULL;
	while (fgets(bufor, MAX_BUFOR, fd)) {
		B = (STUDENT*)malloc(sizeof(STUDENT));
		if (H == (STUDENT*)NULL)
		{
			H = B;
		}
		else
		{
			P->n = B;
		}
		B->n = (STUDENT*)NULL;

		len = (unsigned)strlen(bufor);
		bufor[len - 1] = '\0';
		B->imie = (char*)malloc(len);
		strcpy(B->imie, bufor);

		fgets(bufor, MAX_BUFOR, fd);
		len = (unsigned)strlen(bufor);
		bufor[len - 1] = '\0';
		B->nazwisko = (char*)malloc(len);
		strcpy(B->nazwisko, bufor);

		fgets(bufor, MAX_BUFOR, fd);
		B->rok = atoi(bufor);

		fgets(bufor, MAX_BUFOR, fd);
		len = (unsigned)strlen(bufor);
		bufor[len - 1] = '\0';
		B->adres = (char*)malloc(len);
		strcpy(B->adres, bufor);

		fgets(bufor, MAX_BUFOR, fd);
		B->stypendium = atof(bufor);
		P = B;
	}
}
```

Fragment programu – funkcja main():

```
int main(){
	STUDENT* ms;
	fd = fopen("dane.txt", "r");
	if (fd == NULL) {
		printf("Nie mogę otworzyć pliku dane.txt do odczytu!\n");
		getchar();
		exit(1);
	}
	list1(fd);
	dispList1();
	ms = max_s();
	printf("%s %.2lf\n", ms->nazwisko, ms->stypendium);
	addHeadList1();
	addTailList1();
	dispList1();
	printf("dlugosc listy = %d\n", lenList1());
	remHeadList1();
	dispList1();
	remTailList1();
	dispList1();
	}
```

3. Unia - rodzaj typu danych, który pozwala na różną interpretację zaalokowanego obszaru pamięci. Składowe unii wskazują zawsze na początek zaalokowanego obszaru. Obszar ten ma rozmiar największej składowej unii. Ze względu na różnice w architekturach komputerów jak i samych kompilatorach języka C, typ ten jest nieprzenaszalny, tzn. ten sam program skompilowany przez inny kompilator może dać zupełnie inne wyniki, np. (za Wikipedia):

```
union nazwa_unii {
	char tablica_znakowa[4];
	int  wartosc;
};
u.wartosc = 0x44434241;
```

W `u.tablica_znakowa` znalazł się napis:

- 'ABCD' w przypadku procesora Intel, kompilator 32-bitowy
- 'AB' w przypadku procesora Intel, kompilator 16-bitowy dla DOS-u
- 'DCBA' w przypadku procesora Motorola.

4. Utwórz strukturę składającą się z dwóch pól typu `__int8`. Utwórz unię, której pierwszy element będzie typu struktury a drugi typu `__int16`. Zainicjuj wystąpienie unii i przypisz wartości do pól struktury (`__int8`) z zakresu od 0 do 255 (dlaczego z takiego zakresu?) Wyświetl elementy struktury i element unii (typu `__int16`) w postaci dziesiętnej `"%i\n"` i szesnastkowej: `"%x\n"`. Co zaobserwowałaś/eś? Zapoznaj się z problemem big-endian/little-endian.

```
struct liczba8 {
	__int8 a;
	__int8 b;
};
```

```
union liczba16 {
	struct liczba8 jeden;
	__int16 dwa;
};
```