Електронски факултет у Нишу 30.9.2024. године

Објектно оријентисано програмирање

Октобар II

Опште напомене:

- Атрибути не могу бити јавни.
- Написати све потребне јавне методе које ће обезбедити да програм ради (укључујући конструкторе и деструктор).
- Избећи на сваки начин дуплирање кода у целом пројекту.
- Придржавати се основних принципа објектно оријентисаног програмирања.

Задатак 1:

Студент је одлучио да направи апликацију која ће њему и његовим колегама помоћи у планирању припреме само једног испита. Апликација треба да буде флексибилна и омогући чување података о испитним материјалима везаних за припрему једног испита. Апликација тренутно треба да омогући рад са две врсте испитних материјала, лекцијама и задацима. Сваки испитни материјал описан је ознаком (јединствен број), тежином (реалан број) и статус да ли је испитни материјал савладан или не. Свака лекција додатно је описана дужином у страницама (реалан број). Сваки задатак поред података о испитном материјалу описан је бројем линија кода решења задатка.

Апликација за планирање припреме испита прави колекцију испитних материјала које се налазе у њему. Величина колекције се дефинише при њеном прављењу и може се мењати током рада апликације. Студент може додати у апликацију нови материјал на крај колекције, ако постоји слободно место. Поред тога, он може да се обрише материјал који је погрешно унео на основу задате ознаке, али само ако испитни материјал није савладан. Студент може у сваком тренутку видети списак материјала, при чему за сваки треба исписати све податке које он садржи. Када студент проучи неки материјал може променити статус на савладан на основу задате ознаке. Студент може одредити укупан број материјала које је савладао. Он може да одреди укупно време потребно за савладавање свих материјала на основу задатог процењеног времена потребног за савладавање јединичног материјала (нпр. једна страница лекције) тежине 1. Време потребно за савладавање лекције се одређује као производ дужине, тежине и процењеног времена. Време потребно за савладавање задатка се одређује као производ броја линија, тежине, процењеног времена и коефицијента сразмерности који износи 0.05. Могуће је одредити и вратити два материјала, онај за који је потребно најмање и онај за који је потребно највише времена за савладавање на основу задатог процењеног времена потребног за савладавање јединичног материјала тежине 1 (једним проласком кроз колекцију).

Проверити рад апликације о припреми испита. Прво додати неколико лекција и задатка, затим обрисати један материјал и поставити неки материјал на савладан. Приказати садржај припреме испита након сваке измене. Поред тога одредити укупан број савладаних материјала, укупан време потребно за припрему испита. На крају вратити и приказати све податке о материјалима за које је потребно најмање и највише времена за припрему.

Задатак 2:

Апликација за припрему усменог дела испита из произвољног предмета на студијама води евиденцију о лекцијама које треба савладати. Апликација треба да буде флексибилна како би могла да користи различите типове података који описују лекције и да обезбеди чување произвољног броја ових података. Кориснику ове апликације потребно је омогућити додавање нове лекције на крај колекције (ако је колекција пуна пријавити проблем). За случај да се нека област избаци из предмета могуће је избацити низ суседних лекција из колекције на основу задатог почетног редног броја и броја лекција које треба уклонити (ако не постоји нека од лекција из низа пријавити проблем). Омогућити одређивање и враћање укупне тежине свих лекција у колекцији. Обезбедити одређивање и враћање броја и укупне тежине свих лекција чија тежина је већа од задате вредности једним проласком кроз елементе колекције. Омогућити одређивање и враћање најлакше и најтеже лекције која се налази у колекцији између задатог почетног и крајњег редног броја, једним проласком кроз те елементе у колекцији (пријавити проблем ако су редни бројеви ван колекције). Апликација може да сачува колекцију у датотеку задатог назива за каснију анализу (ако дође до проблема при чувању колекције пријавити исти). Сачуване податке је могуће учитати из датотеке задатог назива у нову колекцију и наставити даљи рад са њом (ако дође до проблема при учитавању података пријавити исти).

Рад апликација за припрему усменог дела испита проверити за случај простих лекција и нормалних лекција. Код простих лекција о свакој лекцији се чува само реална вредност њене тежине. Код нормалних лекција памте се назив (низ карактера), дужину у страницама и процењена тежина. Тежина нормалне лекције се одређује као производ дужине и процењене тежине.

Проверити рад апликације за оба типа испита. Прво треба додати податке у колекцију, затим избацити један низ лекција, После одредити и приказати укупну вредност тежине лекција. Затим израчунати и приказати број и укупну тежину лекција тежих од задате вредности. Након тога одредити и вратити најлакшу и најтежу лекцију између задатих редних бројева. На крају сачувати садржај новчаника у датотеку, учитати их у други новчаник. Приказати све податке из колекције након додавања и брисања, као и након учитавања у нову колекцију. Проширити главни програм тако да обезбеди обраду изузетака.

ПРЕДМЕТНИ НАСТАВНИЦИ И АСИСТЕНТИ

main.cpp

```
void zadatak1()
{
	float vJedinicno = 10;

	// 2 poena
	IspitniMaterijal* p1 = new Lekcija(1001, 2.0, 1.5);
	//p1->Prikazi();
	cout << *p1 << endl;
	// 2 poena
	IspitniMaterijal* p2 = new Zadatak(1002, 2.4, 20);
	//p2->Prikazi();
	cout << *p2 << endl;

	float tezine[] = { 1.5, 2.7, 0.8, 3.2, 1.8 };
	float duzina[] = { 1.2, 3.1, 2.5, 3.4, 4.1 };
	int kod[] = { 100, 20, 40, 50, 30 };
	IspitPriprema* pIspit = new IspitPriprema(20);
	// 2 poena
	for (int i = 0; i < 5; i++)
	{
		Lekcija* lekcija = new Lekcija(2 * i + 1, tezine[i], duzina[i]);
		pIspit->Dodaj(lekcija);
		Zadatak* zadatak = new Zadatak(2 * i + 2, tezine[i], kod[i]);
		pIspit->Dodaj(zadatak);
	}
	// 2 poena
	//pe.print();
	cout << *pIspit << endl;

	// 2 poena
	pIspit->Obrisi(3);
	//pe->print();
	cout << *pIspit << endl;

	// 2 poena
	pIspit->Savladaj(2);
	//pe->print();
	cout << *pIspit << endl;

	// 2 poena
	int sav = pIspit->UkupnoSavladanih();
	cout << sav << endl;

	// 3 poena
	float vreme = pIspit->UkupnoVreme(vJedinicno);
	cout << vreme << endl;

	// 2 poena
	IspitniMaterijal* matMin = nullptr, * matMax = nullptr;
	pIspit->VratiNaj(vJedinicno, &matMin, &matMax);
	cout << *matMin << endl << *matMax << endl;
	//matMin->Prikazi();
	//matMax->Prikazi();

	// 1 poena
	delete pIspit;
}
```

```
void zadatak2() {
	int maxPodataka = 8, brojPodataka = 9;

	// 3 boda
	// postavljanje i prihvatanje izuzetaka
	{
		float tezine[] = { 1.5, 2.7, 3.2, 0.8, 1.8, 4.1, 2.5, 2.3, 3.6 };
		Ispit<float> ispit(maxPodataka);
		// 1 bod
		for (int i = 0; i < brojPodataka; i++)
			ispit.Dodaj(tezine[i]);
		cout << ispit << endl; //nov.Ispisi(cout);
		// 1 bod
		ispit.Izbaci(4, 3);
		cout << ispit << endl; //nov.Ispisi(cout);

		// 1 bod
		float ukupno = ispit.Ukupno();
		cout << ukupno << endl; // 14.1

		// 1 bod
		float vrednost = 2.0; float ukupnaVrednost = 0;
		int broj = ispit.UkupniBrojVrednost(vrednost, ukupnaVrednost);
		cout << broj << " " << ukupnaVrednost << endl; // 4 11.8

		// 1 bod
		float min = 0.0; float max = 1000.0;
		ispit.minMax(1, 4, min, max);
		cout << min << " " << max << endl; // 4

		// 1 bod
		ispit.Sacuvaj("IspProst.txt");
		// 1 boda
		Ispit<int> ispUc(maxPodataka);
		ispUc.Ucitaj("IspProst.txt");
		cout << ispUc << endl; //novUcitan.Ispisi(cout);
	}

	{
		char naziv[][8] = { "prva", "druga", "treca", "cetvrta", "peta",
				    "sesta", "sedma", "osma", "deveta" };
		float tezine[] = { 1.5, 2.7, 3.2, 0.8, 1.8, 4.1, 2.5, 2.3, 3.6 };
		float duzina[] = { 1.2, 3.1, 2.5, 3.4, 4.1, 1.7, 2.2, 3.7, 2.8 };
		Ispit<LekcijaNormalna> ispNor(maxPodataka);
		// 1 bod
		for (int i = 0; i < brojPodataka; i++) {
			LekcijaNormalna lekcija(naziv[i], tezine[i], duzina[i]);
			ispNor.Dodaj(lekcija);
		}
		// 1 bod
		cout << ispNor << endl; //novVal.Ispisi(cout);
		// 1 bodova
		ispNor.Izbaci(4, 3);
		cout << ispNor << endl; //novVal.Ispisi(cout);

		// 1 bod
		float ukupno = ispNor.Ukupno();
		cout << ukupno << endl;

		// 2 bod
		float vrednost = 2.0; float ukupnaVrednost = 0;
		int broj = ispNor.UkupniBrojVrednost(vrednost, ukupnaVrednost);
		cout << broj << " " << ukupnaVrednost << endl;
```

```
		// 2 bod
		LekcijaNormalna min, max;
		ispNor.minMax(1, 4, min, max);
		cout << min << " " << max << endl;

		// 1 bod
		ispNor.Sacuvaj("IspNormal.txt");
		// 1 bod
		Ispit<LekcijaNormalna> ispNorUc(maxPodataka);
		ispNorUc.Ucitaj("IspNormalB.txt");
		cout << ispNorUc << endl; //novValUc.Ispisi(cout);
	}
}

void main()
{
	zadatak1();
	zadatak2();
}
```