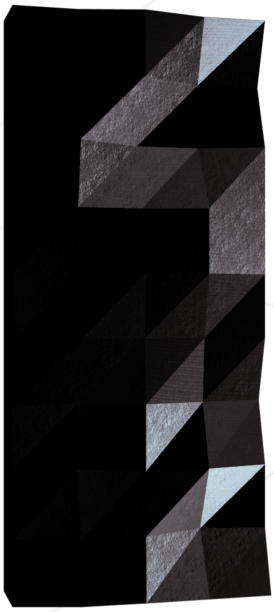

C++ - Մոդուլ 01

Հիշողության բաշխում, Հղումներ, Ցուցիչներ անդամների վրա, Ֆայլային հոսքեր

Հակիրճ. այս փաստաթուղթը C++ -ի 01 մոդուլի նյութն է:

Ցանկ

I	Ընդհանուր կանոններ	2
II	Առաջադրանք 00: Քառոտանիների կույտ	5
III	Առաջադրանք 01: Ծորակների խնդիր	6
IV	Առաջադրանք 02: Ուղեղների քաղհան	7
V	Առաջադրանք 03: Ավելի շատ ուղեղնե՞ր	9
VI	Առաջադրանք 04: ԲԱՐԵՎ, ՍԱ ՈՒՂԵՂ Է	10
VII	Առաջադրանք 05: ԲԱՐԵՎ ՈՒՂԵՂ, ՍԱ ՄԱՐԴ Է	11
VIII	Առաջադրանք 06: Անտեղի բռնություն	12
IX	Առաջադրանք 07: Sed-ը թույլիկների համար է	14
X	Առաջադրանք 08: Չեմ լսել ֆանտաստիկ անջատիչների մասին	15
XI	Առաջադրանք 09: Գերգրանցում	17
XII	Առաջադրանք 10: Ինը ծայրով ճիպոտ	19

Գլուխ I

Ընդհանուր կանոններ

- Յուրաքանչյուր ֆունկցիա, որն իրականացվում է վերնագրում (բացի շաբլոններից) և ցանկացած չպաշտպանված վերնագիր 0 է գնահատվում:
- Յուրաքանչյուր ելք ստանդարտ ելք է և պետք է վերջանա նոր տողով, եթե նշված չէ այլ կերպ:
- Ֆայլերի, ինչպես նաև դասերի, ֆունկցիաների և մեթոդների անունները պետք է տառ առ տառ համապատասխանեն հանձնարարվածին:
- Հիշեք՝ այժմ Դուք ծրագրավորում եք C++-ով, այլ ոչ թե C-ով: Հետևաբար՝
    - հետևյալ ֆունկցիաներն ԱՐԳԵԼՎԱԾ են և դրանց օգտագործումը կգնահատվի 0 միավոր, առանց քննարկման՝ `*alloc`, `*printf` և `free`:
    - Թույլատրվում է ստանդարտ գրադարանից օգտագործել ամեն ինչ: ԻՆՉԵՎԷ, խելամիտ կլինի փորձել և օգտագործել C-ի՝ C++-ին նման ձեզ ծանոթ ֆունկցիաները՝ փոխանակ պարզապես անելու այն, ինչ գիտեք. ի վերջո, C++-ը նոր լեզու է: Նաև ՉԻ թույլատրվում օգտագործել ՍԾԳ-ն (Ստանդարտ Շաբլոնների Գրադարան, STL), քանի դեռ օգտագործման իրական անհրաժեշտություն չկա (այն է՝ մինչև մոդուլ 08-ը): Դա նշանակում է՝ մինչ այդ չօգտագործել վեկտորներ, ցուցակներ, արտապատկերումներ, և այլն... կամ որևէ այլ բան, որը պահանջում է include <algorithm>-ի ներառում:
- Իրականում, բացահայտ կերպով արգելված ցանկացած ֆունկցիայի կամ մեխանիզմի կիրառումը կպատժվի 0-ով, առանց քննարկման:
- Նաև ի նկատի ունեցեք, որ եթե այլապես նշված չէ, C++-ի «using namespace» և «friend» բանալի բառերն արգելված են: Դրանց կիրառումը կպատժվի -42-ով, առանց քննարկման:

- Դասի հետ կապված ֆայլերը պետք է միշտ լինեն `ClassName.hpp` և `ClassName.cpp`, եթե այլապես նշված չէ:
- Հանձնման պահոցներն են՝ `ex00/, ex01/, ..., exn/`
- Պետք է ուշադիր կարդալ օրինակները: Դրանք կարող են պարունակել պահանջներ, որոնք ակնհայտ չեն առաջադրանքների նկարագրության մեջ: Եթե ինչ-որ բան երկիմաստ է թվում, ապա `C++` բավականաչափ չեք հասկանում:
- Քանի որ թույլատրվում է օգտագործել `C++`-ի գործիքները, որոնց ծանոթացել եք հենց սկզբից, չի թույլատրվում օգտագործել որևէ արտաքին գրադարան: Եվ նախքան կհարցնեք, դա նաև նշանակում է, որ չեն թույլատրվում `C++11` ու ածանցյալներ, և ոչ էլ `Boost` կամ որևէ այլ բան, առանց որի `C++`-ը, ըստ ձեր հոյակապ գիտելիքներով ընկերոջ,չի կարող գոյություն ունենալ:
- Ձեզանից կարող է պահանջվել վերադարձնել մեծ քանակով դասեր: Դա կարող է հոգնեցուցիչ թվալ, եթե չկարողանաք հրահանգներ և հրամաններ գրել ձեր սիրելի տեքստային խմբագրում:
- Սկսելուց առաջ ԱՄԲՈՂՋՈՒԹՅԱՄԲ կարդացե՛ք յուրաքանչյուր առաջադրանքը: Իսկապես, արե՛ք դա:
- Պետք է օգտագործել `clang++` կազմարկիչ:
- Ձեր կոդը պետք է կազմարկվի հետևյալ դրոշակների օգնությամբ՝ `-Wall -Wextra -Werror`:
- Ձեր include-ներից յուրաքանչյուրը պետք է կարողանա ներառվել մյուսներից անկախ: Բնականաբար, include-ները պետք է պարունակեն բոլոր այլ ներառումները, որոնցից կախված են:
- Եթե հետաքրքրում է, ասենք, որ `C++`-ում կոդավորման որևէ ոճ չի պարտադրվում: Կարող եք օգտագործել ցանկացած ոճ, որը ձեզ դուր է գալիս, առանց որևէ սահմանափակման: Ինչևէ, հիշեք, որ եթե ձեր գնահատող ընկերը չկարողանա կարդալ կոդը, նա չի կարողանա նաև գնահատել:
- Այժմ ամենակարևորը. դուք ՉԵՔ գնահատվելու ծրագրի միջոցով, եթե դա նյութում բացահայտ կերպով նշված չէ: Հետևաբար, առաջադրանքների կատարման հարցում ձեզ որոշակի ազատություն է տրվում: Այնուամենայնիվ, ուշադիր եղե՛ք յուրաքանչյուր առաջադրանքի սահմանափակումներին, և ՄԻ՛ ծուլացեք, քանի որ առաջադրանքների օգտակար հատկանիշներից շատերը կարող եք բաց թողնել:
- Մի քանի ավել ֆայլ հանձնելը խնդիր չէ, կարող եք նախընտրել ձեր կոդն առանձնացնել ավելի շատ ֆայլերում, քան պահանջվում է: Ձեզ ազատ զգացե՛ք, քանի դեռ արդյունքը ծրագրով չի գնահատվում:

- Նույնիսկ եթե առաջադրանքի նյութը կարճ է, արժե դրա վրա որոշ ժամանակ ծախսել, որպեսզի միանգամայն վստահ լինեք, որ հասկանում եք, թե ինչ է ակնկալվում ձեզնից, և որ դա արել եք հնարավորինս լավագույն կերպով:
- Դե՛, ձեզ տեսնենք: Հանուն Օդինի, հանուն Արամազդի: Ուղեղներդ ի գո՛րծ:

Գլուխ II

Առաջադրանք 00: Քառոտանիների կույտ

	Առաջադրանք 00
Քառոտանիների կույտ	
Հանձնման պահոց՝ $ex00/$	
Հանձնվելիք ֆայլեր՝ Pony.cpp Pony.hpp main.cpp	
Արգելված ֆունկցիաներ՝ ոչ մի	

Սկսենք հեշտից:

Ստեղծել `Pony` դաս, որը պարունակում է այն, ինչ ձեր կարծիքով բավարար կերպով նկարագրում է պոնիներին: Ապա ստեղծել երկու ֆունկցիաներ՝ `ponyOnTheHeap` և `ponyOnTheStack`, որոնցում կզետեղեք `Pony` և նրան ինչ-որ բաներ անել կտաք:

Իհարկե, առաջին `Pony`-ն պետք է զետեղված լինի դինամիկ հիշողության մեջ, երկրորդը՝ սթեքում:

Պետք է ապահովել բավական կոդով `main`, որպեսզի ապացուցեք, որ աշխատանքը կատարված է ինչպես նախատեսված էր:

Երկու դեպքում էլ `Pony` օբյեկտները չպետք է այլևս գոյություն ունենան, երբ ֆունկցիայից դուրս եք գալիս: (Ձեր main-ը պետք է նաև դա ցույց տա:)

Գլուխ III

Առաջադրանք 01: Ծորակների խնդիր

	Առաջադրանք 01
Ծորակների խնդիր	
Հանձնման պահոց՝ $ex01/$	
Հանձնվելիք ֆայլեր՝ ex01.cpp	
Արգելված ֆունկցիաներ՝ ոչ մի	

Եվս մեկ հեշտ առաջադրանք:

Պետք է հանձնել հետևյալ ֆունկցիան՝ դրանում հիշողության արտահոսքը ուղղելուց հետո:

Իհարկե, պետք է խաղալ հիշողության բաշխման, վերաբաշխման հետ: Ուղղակի փոփոխականի հեռացումը, կամ առանց իրական լուծման խնդրի շուրջ ջուր ծեծելը կհամարվի սխալ պատասխան:

```
void        memoryLeak()
{
        std::string*        panther = new std::string("String panther");

        std::cout << *panther << std::endl;
}
```

Գլուխ IV

Առաջադրանք 02: Ուղեղների քաղհան

	Առաջադրանք 02
Ուղեղների քաղհան	
Հանձնման պահոց՝ $ex02/$	
Հանձնվելիք ֆայլեր՝ Zombie.cpp Zombie.hpp ZombieEvent.cpp ZombieEvent.hpp main.cpp	
Արգելված ֆունկցիաներ՝ ոչ մի	

Սկզբի համար ստեղծել `Zombie` դաս: Այն պետք է (առնվազն) պարունակի մեկ տիպ և մեկ անուն, նաև ավելացնել `announce()` անդամ ֆունկցիա, որը կարտածի նման մի բան՝

```
<name (type)> Braiiiiiiinnnssss...
```

Ինչ կամենաք, իրո՛ք, քանի դեռ արտածում եք `Zombie`-ի անունն ու տիպը:

Դրանից հետո ստեղծել `ZombieEvent` դաս: Այն կունենա `setZombieType` ֆունկցիա, որն օբյեկտում կպահի տիպ և `Zombie* newZombie(std::string name)` ֆունկցիա, որը կստեղծի ընտրված տիպով `Zombie`, այն կանվանի և կվերադարձնի:

Ստեղծել նաև `randomChump` ֆունկցիա, որը կստեղծի պատահական անվամբ `Zombie` և կստիպի հայտարարել (announce) ինքն իրեն: Ձեր ընտրած ցանկացած պատահական մեթոդ, իրական պատահական անուններ, կամ անունների ցուցակից պատահական ընտրություն, ընդունելի է:

Պետք է հանձնել ամբողջական ծրագիր՝ ներառյալ `main`-ը, ինչը բավական կլինի ապացուցելու, որ այն ինչ արել եք, աշխատում է ինչպես պետք է: Օրինակ, այնպես անել, որ ձեր նոր սարքած Զոմբիներն իրենք իրենց հայտարարեն:

Այժմ առաջադրանքի հիմնական մասը: Ձեր Զոմբիները պետք է ոչնչացվեն ճիշտ պահին (այսինքն երբ այլևս անհրաժեշտ չեն): Դրանք նաև պետք է բաշխված լինեն նպատակահարմար կերպով. երբեմն նպատակահարմար է, երբ դրանք սթեքում են, երբեմն դինամիկ հիշողությունն ավելի լավ ընտրություն է: Դրական գնահատական ստանալու համար պետք է հիմնավորել, թե ինչ եք արել:

Գլուխ V

Առաջադրանք 03: Ավելի շատ ուղեղնե՞ր

	Առաջադրանք 03
Ավելի շատ ուղեղնե՞ր	
Հանձնման պահոց՝ $ex03/$	
Հանձնվելիք ֆայլեր՝ Zombie.cpp Zombie.hpp ZombieHorde.cpp ZombieHorde.hpp main.cpp	
Արգելված ֆունկցիաներ՝ ոչ մի	

Օգտագործելով նախորդ առաջադրանքում ձեր սարքած `Zombie` դասը՝ ստեղծել `ZombieHorde` դաս:

Այդ դասը պետք է ունենա կոնստրուկտոր, որը ստանում է N ամբողջ թիվ: Ստեղծելիս այն պետք է բաշխի պատահական անուններով («պատահական», ինչպես նախորդ առաջադրանքում) N `Zombie` օբյեկտներ և պահի դրանք: Այնուհետև, այն պետք է ունենա `announce()` ֆունկցիա, որը կանչում է `announce()`-ը յուրաքանչյուր պահված `Zombie`-ի համար:

Բոլոր `Zombie` օբյեկտները պետք է բաշխվեն մեկ բաշխմամբ և ոչնչացվեն, երբ `ZombieHorde`-ը ոչնչացվում է:
Ինչպես միշտ, ներկայացնել `main`-ը թեստերով և հիմնավորել ձեր ընտրությունները:

Գլուխ VI

Առաջադրանք 04: ԲԱՐԵՎ, ՍԱ ՈՒՂԵՂ Է

	Առաջադրանք 04
ԲԱՐԵՎ, ՍԱ ՈՒՂԵՂ Է	
Հանձնման պահոց՝ $ex04/$	
Հանձնվելիք ֆայլեր՝ ex04.cpp	
Արգելված ֆունկցիաներ՝ ոչ մի	

Գրել ծրագիր, որում կստեղծեք `"HI THIS IS BRAIN"` պարունակող տող, ցուցիչ դրա վրա և հղում դրան:

Այնուհետև, պետք է արտածել այն՝ օգտագործելով ցուցիչը և վերջապես, արտածել այն՝ օգտագործելով հղումը:

Ահա և վերջ, ոչ մի բարդ բան կամ հնարք:

Գլուխ VII

Առաջադրանք 05: ԲԱՐԵՎ ՈՒՂԵՂ, ՍԱ ՄԱՐԴ Է

	Առաջադրանք 05
ԲԱՐԵՎ ՈՒՂԵՂ, ՍԱ ՄԱՐԴ Է	
Հանձնման պահոց՝ $ex05/$	
Հանձնվելիք ֆայլեր՝ Brain.cpp Brain.hpp Human.cpp Human.hpp main.cpp	
Արգելված ֆունկցիաներ՝ ոչ մի	

Ստեղծել `Brain` դաս այն ամենով, ինչը կարծում եք սազում է ուղեղին: Այն կունենա `identify()` ֆունկցիա, որը վերադարձնում է հիշողության մեջ ուղեղի հասցեն՝ 0x նախածանցով տասնվեցական հաշվարկման համակարգում (օրինակ՝ «`0x194F87EA`»):

Ապա սարքել `Human` դաս, որն ունի հաստատուն `Brain` հատկանիշ՝ միևնույն կյանքի տևողությամբ: Այն ունի `identify()` ֆունկցիա, որն ուղղակի կանչում է իր `Brain`-ի `identify()` ֆունկցիան և վերադարձնում դրա արժեքը:

Այժմ այնպես անել, որ այդ կոդը կազմարկի և արտածի երկու նույնական հասցեներ՝

```
int main()
{
        Human        bob;

        std::cout << bob.identify() << std::endl;
        std::cout << bob.getBrain().identify() << std::endl;
}
```

Այս կոդը պետք է հանձնվի որպես ձեր `main`, և այն աշխատացնելու համար ինչ որ ավելացվի `Human` կամ `Brain` դասերին, պետք է հիմնավորվի («Էհ, այո, դե այնքան խաղացի դրա հետ, մինչև աշխատեց»-ից տարբերվող փաստարկով):

Գլուխ VIII

Առաջադրանք 06: Անտեղի բռնություն

	Առաջադրանք 06
Անտեղի բռնություն	
Հանձնման պահոց՝ $ex06/$	
Հանձնվելիք ֆայլեր՝ Weapon.cpp Weapon.hpp HumanA.cpp HumanA.hpp HumanB.cpp HumanB.hpp main.cpp	
Արգելված ֆունկցիաներ՝ ոչ մի	

Սարքել `Weapon` դաս, որն ունի մեկ `type` տող և մեկ `getType`, որը վերադարձնում է const հղում այդ տողին: Նաև ունի `setType`:

Այժմ ստեղծել երկու երկու `HumanA` և `HumanB` դասեր, որոնք երկուսն էլ ունեն մեկ `Weapon`, անուն և մեկ `attack()` ֆունկցիա, որն արտածում է նման մի բան՝

```
NAME attacks with his WEAPON_TYPE
```

Այնպես անել, որ հետևյալ կոդը «crude spiked club»-ով, ԱՅՆՈՒՀԵՏԵՎ՝ «some other type of club»-ով հարձակումներ առաջացնի՝ երկու թեստերի դեպքում էլ՝

```
int main()
{
	{
		Weapon		club = Weapon("crude spiked club");

		HumanA bob("Bob", club);
		bob.attack();
		club.setType("some other type of club");
		bob.attack();
	}
	{
		Weapon		club = Weapon("crude spiked club");

		HumanB jim("Jim");
		jim.setWeapon(club);
		jim.attack();
		club.setType("some other type of club");
		jim.attack();
	}
}
```

Ո՞ր դեպքում է նպատակահարմար պահել `Weapon`-ը որպես ցուցիչ: Իսկ որպես հղո՞ւմ: Ինչո՞ւ: Դա ամենալա՞վ ընտրությունն է՝ հաշվի առնելով ինչ խնդիր է դրված: Սրանք այն հարցերն են, որոնք պետք է ինքներդ ձեզ տաք՝ նախքան առաջադրանքը հանձնելը:

Գլուխ IX

Առաջադրանք 07: Sed-ը թույլիկների համար է

	Առաջադրանք 07
Sed-ը թույլիկների համար	
Հանձնման պահոց՝ $ex07/$	
Հանձնվելիք ֆայլեր՝ Makefile, և այն ամենն ինչ ձեզ պետք է	
Արգելված ֆունկցիաներ՝ ոչ մի	

Գրել `replace` կոչվող ծրագիր, որն ընդունում է ֆայլի անուն և երկու տող, դիցուք՝ s1 և s2 անուններով, որոնք դատարկ ՉԵՆ:

Այն կբացի ֆայլը և դրա պարունակությունը կգրի FILENAME.replace-ում՝ նախապես փոխարինելով բոլոր s1-երը s2-ով:

Իհարկե, սխալները պետք է մշակել հնարավորինս լավ կերպով և չօգտագործել C-ի ֆայլի մանիպուլյատիվ ֆունկցիաներ, քանի որ դա խարդախություն կլինի, իսկ խարդախությունը լավ բան չէ, չէ՞:

Պետք է հանձնել որոշ թեստային ֆայլեր՝ ցույց տալու համար, որ ձեր ծրագիրն աշխատում է:

Գլուխ X

Առաջադրանք 08: Չեմ լսել ֆանտաստիկ անջատիչների մասին

	Առաջադրանք 08
Չեմ լսել ֆանտաստիկ անջատիչների մասին	
Հանձնման պահոց՝ $ex08/$	
Հանձնվելիք ֆայլեր՝ Human.hpp Human.cpp main.cpp	
Արգելված ֆունկցիաներ՝ ոչ մի	

Այս առաջադրանքը միավորներ չի նախատեսում, բայց միևնույնն է, այն օգտակար է: Կարող եք այն կատարել ըստ ձեր ցանկության:

Օգտագործել հետևյալ `Human` դասը՝

```
class Human
{
        private:
        void        meleeAttack(std::string const & target);
        void        rangedAttack(std::string const & target);
        void        intimidatingShout(std::string const & target);

        public:
        void        action(std::string const & action_name, std::string const & target);
};
```

Իրականացնել այս բոլոր ֆունկցիաները, առաջին երեքն ուղղակի ստանդարտ ելքով ինչ-որ բան ցույց կտան, որպեսզի երևա, որ դրանք կանչվել են: Վերջինը պետք է կանչի համապատասխան գործողությունը համապատասխան թիրախի (target) վրա: Պետք է

օգտագործել անդամների ցուցիչների զանգված, որպեսզի ընտրվի, թե որ ֆունկցիան է կանչվելու. բազմակի `if`-երի կամ `switch`-երի կիրառությունն արգելված է:

Գլուխ XI

Առաջադրանք 09: Գերգրանցում

	Առաջադրանք 09
Գերգրանցում	
Հանձնման պահոց՝ $ex09/$	
Հանձնվելիք ֆայլեր՝ Logger.cpp Logger.hpp main.cpp	
Արգելված ֆունկցիաներ՝ ոչ մի	

Այս առաջադրանքը միավորներ չի նախատեսում, բայց միևնույնն է, այն օգտակար է: Կարող եք այն կատարել ըստ ձեր ցանկության:

Ստեղծել `Logger` կոչվող դաս, որը դե պետք է որոշ գրանցումներ անի:

Այն կունենա երկու փակ ֆունցկիաներ՝ `logToConsole` և `logToFile`, որոնք երկուսն էլ ընդունում են տող և այն, համապատասխանաբար, գրում են ստանդարտ ելքում և ավելացնում են այն ֆայլում, որի անունը ստեղծման պահին կպահվի `Logger`-ում:

Ստեղծել նաև `makeLogEntry` կոչվող փակ ֆունկցիա, որը որպես տող կընդունի սովորական հաղորդագրություն և կվերադարձնի որպես նոր տող՝ պարունակվող հաղորդագրությունը ֆորմատավորելով որպես ճիշտ գրանցման տող: Հաղորդագրությունից առաջ պետք է գոնե ավելացնել տվյալ ամիս-ամսաթիվը, այնպես որ հնարավոր լինի տեսնել, թե երբ է գրանցվել:

Վերջապես, ստեղծել `log(std::string const & dest, std::string const & message)`, որը հաղորդագրությունից կկառուցի գրանցում և այն կփոխանցի `logToFile`-ին կամ `logToConsole`-ին՝ կախված `dest`

պարամետրից: Ինչպես և նախորդ առաջադրանքում, պետք է օգտագործել անդամների ցուցիչներ, որպեսզի ընտրվի, թե որ ֆունկցիան է կանչվելու:

Գլուխ XII

Առաջադրանք 10: Ինը ծայրով ճիպոտ

	Առաջադրանք 10
Ինը ծայրով ճիպոտ	
Հանձնման պահոց՝ $ex10/$	
Հանձնվելիք ֆայլեր՝ main.cpp և ինչ որ ձեզ պետք է	
Արգելված ֆունկցիաներ՝ ոչ մի	

Այս առաջադրանքը միավորներ չի նախատեսում, բայց միևնույնն է, այն օգտակար է: Կարող եք այն կատարել ըստ ձեր ցանկության:

Սարքել `cato9tails` ծրագիր, որն անում է նույնը, ինչ համակարգի `cat` հրամանը՝ առանց տարբերակների (options): Այն կարող է կարդալ ֆայլերից և/կամ ստանդարտ ելքից: Թեստավորման ժամանակ եղե՛ք ուշադիր, սա այնքան էլ հեշտ չէ, որքան թվում է: