Домаћи

Пре него што почнете прочитајте поглавље УПУТСВО на крају овог документа. Ту ћете пронаћи упутства како да попуните овај документ и како треба да доставите своја решења задатака из овог домаћег.

**ЗАДАЦИ**

1. Прочитајте поглавља 2 и 3 књиге Бјарнеа Строуструпа „*Programming Principles and Practices C++"*. Научите значења свих појмова из тих поглавља одговорите на следећа питања:
   1. Набројте четири дела дефиниције функције:

1. povratni tip

2. naziv

3. parametri

4. telo

* 1. Приликом учитавања са стандардног улаза, шта означава крај једног уноса за целе бројеве и знаковне низове?

|  |
| --- |
| bilo kakav whitespace (tab, newline, space, EOL) |

* 1. Шта је објекат?

|  |
| --- |
| Objekat je neka memorija koja sardzi vrednost nekog definisanog tipa. |

* 1. Шта је литерал? Наведите неколико типова литерала.

|  |
| --- |
| Literal je neka konstanta koja je definisana u data sekciji programa tj. prisutna je tokom celog programa. Primeri:  "abc", 123, 5.5, 'b', true, ... |

* 1. Шта су променљиве? Наведите основне типове променљивих и њихове уобичајене величине.

|  |
| --- |
| Promenjiva je imenovani pojam u programu u kom se nalazi vrednost neke memorije.  Osnovni tipovi: int (sve varijante od 2, 4 i 8 bajtova), double (8 bajtova), char (1 bajt), bool (1 bajt) |

* 1. Која је разлика између операција = и ==? Ког типа је њихов резултат?

|  |
| --- |
| Operacija = je operacija dodeljivanja i njena povratna vrednost je vrednost izraza sa desne strane. Operacija == je operacija jednakosti i njena povratna vrednost je tipa bool (true oznacava da su levi i desni operand jednaki a false oznacava da  su razliciti). |

* 1. Шта је дефиниција, а шта декларација?

|  |
| --- |
| Definicija zauzima memoriju za objekat i dodeljuje mu ime ili u slucaju funkcije definise i telo. Deklaracija samo govori kompajleru da postoji neki objekat, ili funkcija, pod tim nazivom. |

* 1. Шта је иницијализација и по чему се разликује од доделе?

|  |
| --- |
| Inicijalizacija je kada pri definiciji novog objekta odma dodelimo i vrednost. Svaka sledeca dodela je samo dodela, a ne inicijalizacija. |

* 1. Како се знаковни низови надовезују у Це++ језику?

|  |
| --- |
| Znakovni nizovi u smislu stringa se mogu nadovezati sa + operatorom, a da bi se nadovezali nizovi znakova (char a[]), mora se napraviti nov niz pa se prvo iskopirati prvi niz pa onda drugi. |

* 1. Шта је типска безбедност и зашто је важна?

|  |
| --- |
| Tipska bezbednost oznacava da mi objekte trebamo da koristimo u kontekstu njihovog tipa. Vazna je zato sto ako je ne postujemo, moze doci do nedefinisanog ponasanja ili gubljenja podataka. |

* 1. Да ли је смештање вредности **double** типа у променљиву **int** типа типски безбедно? Зашто?

|  |
| --- |
| Kada smestamo double u int, gubimo sve stvari koje su potencionalno iza zareza u double broju. Int i double imaju drugaciju internu reprezentaciju pa konverzija iz veceg (double) u manji (int) nije bezbedno podrzana. |

1. Напишите добро структуиран и коментарисан Це++ програм који садржи три функције описане у наставку. Када се овде каже „хексадецимални број“ мисли се на стринг са хексадецималним записом броја.
   * 1. Функција 1 – прима неозначени цео број и враћа стринг са записом тог броја у хексадецималној форми. Функција треба да баци изузетак у случају неодговарајућег улаза.
     2. Функција 2 – прима два хексадецимална броја (стрингове са хексадецималним записом броја) и враћа њихов збор исто у хексадецималном облику. Функција треба да баци изузетак у случају неодговарајућег улаза.
     3. Функција 3 – прима хексадецимални број и враћа целобројну вредност која одговара том хексадецималном запису. Функција треба да баци изузетак у случају неодговарајућег улаза.
     4. Позовите ове функције из мејн функције. Употребите трај (try) и кеч (catch) механизме.
     5. Проверите исправност вашег програма помоћу примера:
        1. исправног улаза
        2. неисправног улаза
        3. празног улаза

1. Напишите програм који очекује од корисника да унесе две вредности истог основног типа (**short**, **int** или **double**). Те вредности треба сместити у променљиве **val1** и **val2**. Затим треба одредити мањи и већи од та два броја, њихов збир, разлику, производ и количник, и исписати те резултате на стандардни излаз. Употребите три функције, по једну за сваки тип променљивих. Омогућити кориснику да изабере коју од те три функције (та три типа) жели.

1. Напиши програм који рачуна Фибоначијеве бројеве. Формула за генерисање Фибоначијевих бројева дата је у наставку:

|  |
| --- |
| Fn = Fn-1 + Fn-2, где је F1 = 1 и F2 = 2.  На пример, F3 = F2 + F1 = 2 + 1 = 3. |

Приметите да ће у једном тренутку Фибоначијеви бројеви бити превелики за **int** (или било који други) тип и доћи ће до прекорачења. Размислите како да прекинете рачунање бројева када дође до прекорачења, или непосредно пре него што до прекорачења дође, и то имплементирајте.

Напишите три посебне функције за три различита типа: **int**, **long** и **long** **long**.

**Питања**:

* 1. Колико првих Фибоначијевих бројева можемо сместити у тип **int**? Која је вредност највећег?

|  |
| --- |
| Najveci fibonacijev broj koji mozemo smestiti u int je: 1,836,311,903 (46. po redu) |

* 1. Колико првих Фибоначијевих бројева можемо сместити у тип **long**? Која је вредност највећег?

|  |
| --- |
| Najveci fibonacijev broj koji mozemo smestiti u long je: 1,836,311,903 (46. po redu) |

* 1. Колико првих Фибоначијевих бројева можемо сместити у тип **long long**? Која је вредност највећег?

|  |
| --- |
| Najveci fibonacijev broj koji mozemo smestiti u long long je: 7,540,113,804,746,346,429 (92. po redu) int i long imaju isti opseg na win\_x64 masinama |

**УПУТСТВО:**

Напишите одговоре у одговарајућа поља. Проширите величину поља ако вам треба више места. Решења за задатке II, III и IV, треба да буде приложено уз овај документ. Све то спакујте у Зип архиву. Архиву треба назвати овако:

***ИМЕ\_ПРЕЗИМЕ\_ИНДЕКС\_domaci.zip***

(Где *ИМЕ,* *ПРЕЗИМЕ* и *ИНДЕКС* треба заменити са вашим именом, презименом и бројем индекс)

Архива треба да садржи:

Датотеке: **domaci.docx** (са одговорима на питања)

Директоријуме:

1. – који садржи пројекат и изворни код за задатак 2
2. – који садржи пројекат и изворни код за задатак 3
3. – који садржи пројекат и изворни код за задатак 4

У решењима задатака треба користити само оне елементе језика Це++ које смо спомињали током прве три недеље на предавањима. Циљ није да се размећете напредним знањем Це++ синтаксе.

|  |
| --- |
| **ВАЖНО**  Пројектни директоријуми (*II*, *III* и *IV*) **НЕ СМЕЈУ** садржати извршне или објектне датотеке!!! Дозвољене су искључиво следеће датотеке:  .vcxproj  .sln  .filters  .cpp  .h  .hpp  **Изоставити** аутоматски направљене директоријуме: *Debug*, *Release*, *ipch* и сл. |