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. тип вриједности коју функција враћа

2. име функције

3. листа параметара који јој се прослијеђују

4. тијело функције

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

|  |
| --- |
| Whitespace, tab, newline, endofline или enter. |

* 1. Шта је објекат?

|  |
| --- |
| Објекат је дио меморије дефинисан одређеним типом вриједности, при чему тај тип одређује каква врста информације може бити написана у њега. |

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

|  |
| --- |
| Литерал је константна вриједност која је додијељена некој промјенљивој током кода. У C++ постоји 5 типова литерала: integer, float, character, string, boolean литерал. |

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

|  |
| --- |
| Промјенљиве представљају објекте који имају име и тип који одређује која вриједност може бити смјештена у тај објекат. Основни типови промјенљивих су:  int – 4 бајта (int short – 2 бајта)  bool – 1 бајт  char – 1 бајт  double – 8 бајтова |

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

|  |
| --- |
| Операција = јесте операција додјеле, тј. додјељује нову вриједност промјенљивој и не враћа никакву вриједност, а операција == провјерава да ли су вриједности једнаке или не и враћа bool вриједност (true - ако јесу исте, false - ако нису исте). |

* 1. Шта је дефиниција, а шта декларација?

|  |
| --- |
| Декларација је исказ у коме се неком објекту даје име и даје компајлеру до знања да тај објекат постоји, а дефиниција је декларација која издваја меморијску локацију за неки објекат, и тада компајлер зна да тај објекат постоји и гдје се налази у меморији. |

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

|  |
| --- |
| Иницијализација јесте додјељивање почетне вриједности некој промјенљивој приликом њеног креирања и обавља се само једном.  Додјела се користи за промјену вриједности промјенљивој након што је иницијализована и може се понављати више пута у коду. |

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

|  |
| --- |
| Знаковни низови се надовезују користећи оператор + или функцију за спајање низова – конкатенација. |

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

|  |
| --- |
| Типска безбједност је идеал и генерално правило неког програмског језика које омогућава провјеру компајлера да ли се исправно користе типови података у коду. Идеал је да се не користе могућности језика за које компајлер не може тврдити да су сигурне (статичка типска безбједност). Значај овога јесте што су грешке мање, боља је читљивост и перформансе. |

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

|  |
| --- |
| То није типски безбједно јер вриједности које добијемо том конверзијом могу бити у великој мјери различите од оригиналне вриједности. |

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?

|  |
| --- |
| Можемо смјестити првих 46 бројева у **int** тип, а највећи је 1836311903. |

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

|  |
| --- |
| Можемо смјестити првих 46 бројева у **long** тип, а највећи је 1836311903. |

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

|  |
| --- |
| Можемо смјестити првих 92 броја у **long** **long** тип, а највећи је 7540113804746346429. |

**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. |