Progvo

Dokumentado (skizo)

Versio 0.9α (skizo), 11–a de novembro, 2019

Thomas Nguyen (Pttn)

Progvo.dev

Enhavo

1. Antaŭparolo

1.1 Tiu dokumento

Ĝi estas la oficiala dokumentado de Progvo, elŝutebla en ĝia retejo.

`https://Progvo.dev/Dok/Dokumentado.pdf`

Lerniloj kaj ekzemploj ne estas en la kampo de la dokumentado, vidi anstataŭe aliajn dokumentojn, kiel la oficiala retejo de la projekto `https://Progvo.dev/Lerni/`.

Dankon por via intereso por Progvo!

1.2 Pri Progvo

Progvo estas ĝeneralcela programlingvo, kreata per Thomas Nguyen (Pttn) en 2019[2]. Ĝi estas nova projekto, kaj la longtempe celo estas havi simplan, diversutilan kaj produkteman programlingvon.

Ĝi ne havas striktan paradigmon, sed havas multe atributojn de objektemaj, funkciaj kaj proceduraj programlingvoj.

Programaroj skribata en Progvo devas dependi la malplej ebla de la sistemo, do ĝi estas altnivela programlingvo. Tamen, malaltnivelaj kapabloj povas ankaŭ esti disponebla en kelkaj da realigoj de Progvo.

1.3 Publiko

Progvo devus esti uzebla per ajn programisto kaj sufice simpla por novuloj, ĝi povas esti amuza unua programlingvo por novaj programistoj. Provu ĝin!

1.4 Realigoj

Realigoj de Progvo devas sekvi la dokumentadon. Alie, ĝi ne estas pli de kondiĉoj, kaj realigoj povas ekzemple esti Progvo interpretilo aŭ programtradukilo.

Ili povas ankaŭ realigi pli de *eksteraj* klasoj kaj funcioj ol tiuj en la dokumentado, sed ne devas permesi novajn aŭ malsamajn sintaksojn aŭ semantikojn.

[1] `https://creativecommons.org/licenses/by-sa/4.0/legalcode`

[2] Unua versio de la dokumentado estas publikigita la 11-a de Novembro 2019.

2. Fundamento

2.1 Lingvo

2.1.1 Alfabeto kaj vortoj

Progvo uzas la signara kodoprezento *Ekio*[1]. La *alfabeto* Σ de la programlingvo estas:

- Spaco, tabo kaj linisalto (Σ_{spa});
- Ciferoj (Σ_{cif}): `0123456789`;
- Literoj (Σ_{lit}): `abcĉdefgĝhĥijĵklmnoprsŝtuŭvzqwxyABCĈDEFGĜHĤIJĴKLMNOPRSŜTUŬVZQWXY`;
- Limigiloj (Σ_{lim}): `↓↑↤↦|/\,;.:[]{}`;
- Aliaj specialaj signoj (Σ_{spe}): `+ − ×÷ = ¬∨∧ ⇔ ∞ <> ∅ ∈*°′ ∫^~±ΔΠ○%‰'"!?§#@★-_`.

Se la kodon havas signon $s \notin \Sigma$, ĝi estas nevalida kaj nenian programaron povas esti kreata (subteno de UTF-8 kaj aliaj povas veni en poste versioj de Progvo).

2.1.2 Komentoj

Oni *komentas* la kodon uzanta la simbolon `§`. Se `§` estas sekvata per `[`, `§` kaj signoj poste ĝi ĝis `]` aŭ la fino de la dosiero estas ignorata.

Alie, `§` kaj signoj poste ĝi ĝis la sekva linisalto aŭ la fino de la dosiero estas ignorata.

2.1.3 Ĵetonoj

Ĵetonoj estas ŝlosilvortoj, limigiloj, identigiloj kaj konstantoj. Ili estas apartiginta kun spaco(j), sed limigiloj povas ankaŭ esti algluata kun aliaj ĵetonoj.

Taboj kaj linisaltoj estas rigardata kiel spacoj. Multe spacoj estas sama ol nur unu spaco.

Ŝlosilvortoj

Progvo havas sekvajn *ŝlosilvortojn* (aro $\hat{S}$): `ERARO`, `INKLUZIVI`, `FINO`, `NOMSPACO`, `EKSTERA`, `KLASO`, `KONSTANTO`, `KONSTRUILO`, `SE`, `ALIE`, `RIPETI`, `DUM`, `HALTI`, `ELIRI`.

Ili estas usklecodistinga.

Limigiloj

Limigiloj (aro L) estas unuopaj simboloj de Σ_{lim}. Ekzemple `.`, `;`, `{`, `)`, `↦`.

Identigiloj

Identigiloj estas $I = ((\Sigma \setminus (\Sigma_{lim} \cup \Sigma_{spa}))^+ \setminus (\hat{S} \cup K)$. Ekzemple `abc`, `+`, `a1b2c3`, `7a8b9c`, sed ne ϵ (malplena vorto), `.`, `[]`, `SE`, `↦`, `7a8 b9c`, aŭ `a↦b`.

[1] Vidi `https://progvo.dev/Ekio.html` por la oficiala tabelo.

Konstantoj

Konstantoj (aro K) estas spacialaj vortoj, prezentanta konstantajn objektojn de klaso difinita en la dokumentado.

2.2 Fundamentaj konceptoj kaj sintakso

Ekde nun, la kodo devas esti konvertita en ĵetonoj por la sintaksa analizo.

Por la simpleco, la sintakso estos malformale difinita en ĉiu sekcio, sed devus esti sufice akurata por eviti la plursencecojn. Frazoj ĉirkaŭata kun < kaj > *devas* (inkluzivanta < kaj >) esti anstataŭata kun valida(j) ĵetono(j) indikita(j) per la frazo, tiuj ĉirkaŭata kun `[` kaj `]` *povas* esti anstataŭata kun konvena(j) ĵetono(j).

2.2.1 Instrukcioj kaj sekvencoj de instrukcioj

En Progvo, la koncepto de *instrukcio* estas tre ĝenerala kaj instrukcio povas esti multe aĵoj kiel deklaroj, difinoj aŭ taksadoj. Grupo de ordonaj instrukcioj (kiu povas esti malplena) estas *sekvenco de instrukcioj*, kiu mem estas ankaŭ instrukcio.

Ĉiu instrukcio devas fini kun punktokomo `;`, escepte de sekvencoj de instrukcioj, kiu devas esti ĉirkaŭata kun `{` kaj `}` (escepte la ĉefsekvenco, kiu ne apartenas en alia sekvenco).

Deklaroj kaj difinoj en sekvenco de instrukcioj povas esti uzanta nur en la sama sekvenco (kaj ĝiaj subsekvencoj).

2.2.2 Esprimoj

Esprimo estas grupo de ĵetonoj, kiu povas taksi sin en objekto aŭ listo de objektoj. Vidi malsupre por la difinoj de objektoj kaj kiel la taksado funkcias.

Esprimoj ne estas instrukcioj, sed la taksado de plena esprimo kiu ne estas parto de alia instrukcio estas instrukcio.

2.2.3 Listoj

Listoj estas nur oportuna sintakso, kiu permesas inter aliaj la uzo de funkcioj kun eligoj de multe objektoj

Ili estas ingitaj, kaj la ĝenerala sintakso estas:

- Malprena listo: `()`, ne povas esti en listo de profundeco ≥ 1 (ekzemple `((a, b), (), c)` ne estas valida listo);
- Listo de profundeco 1: `(<ero>[, <ero>[, ...[, <ero>]...]])`;
- Listo de profundeco > 1: `(<listo>[, <listo>[, ...[, <listo>]...]])`.

Listoj ne estas instrukcioj.

Listoj de objektoj

`ero` estas identigilo de objekto.

Listoj de enigo/eligo

`ero` estas deklaro de objekto (sen atribuo) (vidi malsupre).

Listoj de esprimoj

`ero` estas esprimo.

2.2.4 Objektoj

Objekto estas instanco de klaso (vidi malsupre), kiu havas datumojn (staton) kaj metodojn, laŭ ĝia klaso.

Deklaroj kaj difinoj de objektoj estas instrukcioj.

Deklaroj

Oni povas krei nomajn objektojn uzanta *deklaro*jn, kun la sintakso

```
<identigilo (de la klaso)> <identigilo (de la objekto)>
```

Ĉi tiu kreas novan objekton de la provizata klaso.

Atribuoj

Atribuo estas anstataŭi la staton de objekto en tiu de alia objekto. Oni atribuas objekto(j)n kun

```
<identigilo (de la objekto)> ← <kongrua esprimo>
```

aŭ

```
<listo de objektoj> ← <kongrua esprimo>
```

La esprimo devas taksi en (listo de) objekto(j) de la sama klaso(j) ol la objekto(j).

Oni povas ankaŭ ambaŭ deklari kaj atribui objekton en unu instrukcio kun

```
<deklaro de objekto> ← <kongrua esprimo>
```

La objekto povas esti konstanta:

```
KONSTANTO <deklaro de objekto>[ ← <kongrua esprimo>]
```

2.2.5 Referencoj

Referenco estas spaciala objekto de klaso `K'` ("`K` Ref"), kiu permesas la plenan atingon de objekto de klaso `K` kun alinomo. Oni povas krei referencojn kiel normala objektoj, kun la sama sintakso.

La klasoj por referencoj estas aŭtomate deklarata kun iliaj normalaj klasoj, ne bezonas difinojn, kaj estas unikaj.

2.2.6 Klasoj

Klaso estas ŝablono prezentanta aron de objektoj kun komunaj datumtipoj (atributoj) kaj funkcioj (metodoj).

Klasojn proponata per la realigo de Progvo, kiu ne estas tute difinata en la fontkodo estas *eksteraj klasoj*. Objektoj de ekstera klaso havas staton, kiu povas ne nur dependi de ĝia atributoj, sed ankaŭ de la realigo de Progvo; la dependeco devas esti tuta por aliaj klasoj.

Deklaroj kaj difinoj

Oni devas deklari eksterajn klasojn antaŭ de uzi ilin, kun la sintakso

```
EKSTERA KLASO <identigilo>
```

Tuj post kiam ekstera klaso estas deklarita, oni povas uzi ĝin kiel ĉiuj klasoj.

La difino de novaj klasoj (klasoj de uzanto) startas kun `KLASO Klasnomo :` kaj finis kun `FINO KLASO`. En la difino, oni difinas la atributoj kiel deklaroj (sen atribuo) de objektoj, kaj metodoj kun sama sintakso ol normalaj funkcioj.

Deklaroj kaj difinoj de klasoj estas instrukcioj.

Konstruiloj

Konstruilo estas spaciala metodo, kiu kreas novan objekton de la klaso. La sintakso estas

```
KONSTRUILO <listo de enigo> <sekvencoj de instrukcioj>
```

Uzo

Oni acesas atributojn (la enkapsuligo venos en poste versioj de Progvo) kun

```
<identigilo (de la objekto)>.<identigilo (de la atributo)>
```

Metodoj estas uzata kun

```
<identigilo (de la objekto)>.<voko de funkcio>
```

Konstruilojn estas uzata kiel normalaj funkcioj, kies nomo estas la klasnomo. La eligo estas objekto de la klaso.

2.2.7 Funkcioj

Funkcio estas strukturo kun enigo, eligo kaj instrukcioj, kiu povas esti alvokata. La eligo estas la rezulto de la taksado de la alvoko de funkcio kaj dependas de la enigo kaj la instrukcioj. Funkcioj devas esti difinita antaŭe de alvoki ilin, kun la sintakso

`<identigilo> : <listo de enigo> → <listo de eligo> <sekvenco de instrukcioj>`

Difinoj de funkcioj estas instrukcioj. En la sekvenco de instrukcioj, oni povas eliri de la funkcio kun la instrukcio `ELIRI`.

Oni alvokas funkcion kun `<identigilo (de la funkcio)> <listo de esprimoj>`.

2.2.8 Stirstrukturoj

Seoj

Seo permesas de plenumi instrukciojn nur se kondiĉo estas verigata. La sintakso estas

```
SE <esprimo 1> <sekvenco de instrukcioj 1>
[ALIE SE <esprimo 2> <sekvenco de instrukcioj 2>
[ALIE SE <esprimo 3> <sekvenco de instrukcioj 3>
[...
[ALIE SE <esprimo n> <sekvenco de instrukcioj n>]...]]]
```

`[ALIE <sekvenco de instrukcioj n + 1>]`

Ĉiu esprimo devas taksi en objekto de la ekstera klaso `Buleo` (vera aŭ malvera).

La i-a sekvenco de instrukcioj estas plenumata se, kaj nur se, la i-a esprimo taksas en vera kaj $\forall j < i$: la j-a esprimoj en malvera. La esprimo por `ALIE` estas ĉiam vera.

Seoj ne estas instrukcioj.

Iteracioj

Iteracio permesas de ripeti instrukciojn dum kondiĉo estas verigata. La sintakso estas

`RIPETI [DUM <esprimo>] <sekvenco de instrukcioj>`

La esprimo devas taksi en objekto de klaso `Buleo` kaj la sekvenco de instrukcioj estas ripetata dum ĝi estas vera. Kun nur `RIPETI`, la esprimo estas ĉiam vera.

Oni povas haltigi iteracion kun la instrukcio `HALTI`.

2.2.9 Nomspacoj

Oni povas grupi klasojn kaj funkcioj kun *nomspacoj*.

Nomspaco startas kun `NOMSPACO <identigilo>`, finas kun `FINO NOMSPACO` kaj ne estas instrukcio. Nomspacoj povas esti ingitaj.

Ĉiu klaso kaj funkcio difininta en nomspaco `N` devas esti uzata kun la prefikso `N/`. Se `N` estas mem en nomspaco `M`, la prefikso estas `M/N/`.

En nomspacoj, oni povas uzi la prefikson `/` por la deklaroj kaj difinoj: ili estas traktata kvazaŭ ili estis ekstera de la nomspaco. En aliaj kasoj, tio permesas la uzon de klasoj aŭ funkcioj ekstera de la kuranta nompaco.

2.2.10 Tabeloj

Venos en poste versio de Progvo.

2.2.11 Ŝablonoj

Venos en poste versio de Progvo.

2.3 Semantiko

2.3.1 Medioj

Medio estas aro de klasoj, funkcioj kaj objektoj. Ĉiu sekvenco de instrukcioj havas ĝian propran medion, kiu enhavas ankaŭ la erojn de la medioj de patraj sekvencoj.

En sekvenco de instrukcioj, ĉiu fojo ke klaso, funkcio aŭ objekto estas difininta aŭ deklarinta, ĝi estas aldonata al la medio. Du eroj de sama medio ne povas havi saman nomon (Progvo ne havas nomeklipsojn) kaj oni ne povas uzi klasojn kaj objektojn, aŭ alvoki funkciojn, kiu ne estas en la medio. Escepto, du funcioj povas havi saman nomon se ilia enigoj havas malsamajn klasojn.

2.3.2 Daŭrigotaj deklaroj

Klasoj de uzanto povas esti deklarata sen difino, ekzemple por cirklaj dependecoj. La sintakso estas sama ol la deklaro de ekstera klaso, sen la ŝlosilvorto `EKSTERA`.

En ĉiu kaso, la klaso devas esti difinata poste kaj la kontrolo de uzoj de tiuj klasoj estas farata al la fino de la traktado de la fontkodo.

2.3.3 Taksadoj de esprimoj

La *taksadoj* de esprimoj funkcias kiel ĉi tiu:

- La objektoj kaj listoj de objektoj estas bazaj elementoj de esprimoj;
- La konstantoj, se la taŭga klaso estas proponita per la realigo de Progvo, taksas en la konforma objekto;
- La alvokoj de funkcioj estas taksata kiel matematikaj funkcioj (la funkcioj prenas la taksado de iliaj enigoj kaj revenigas objektojn aŭ listojn de objektoj);
- Neniu alia ne povas esti taksinta, kaj ne povas esti esprimo aŭ parte de esprimo.

2.3.4 Reguloj de referencoj

Oni povas krei referencojn, kiu referencas neniun objekton; ĉi tiu estas *malpravalorizinta referenco*.

Referenco ne povas referenci al alia referenco, kaj oni ne povas ŝangi la referencata objekton (sed malpravalorizinta referenco povas referenci objekton poste).

Se la referenco kaj la referencita objekto ne estas konstantaj, oni povas atribui la referencata objekton kun la sama sintakso ol la atribuo de normalaj objektoj (por la kongruo de la esprimo, la referencata objekto estas uzata, ne la referenco). Malpravalorizinta referenco referencos la novan objekton.

2.4 Traktado de la fontkodo kaj plenumo

2.4.1 Medivariabloj

La realigo de Progvo devas havi ĉi tiujn medivariablojn, kiu permesas la programiston de adapti la konduton de la traktado de la fontkodo:

- `INKL`: la defaŭlta dosierujo por la inkluzivendaj dosieroj.

2.4.2 Inkluzivigoj

Moduleca programado estas ebla kun inkluzivigoj de kodo, uzanta

```
INKLUZIVI <Bitsekvenco>
```

La bitsekvenco devas reprezenti la relativan dosierindikon (al la nuna dosiero) en Ekio.

Se la dosierindiko startas kun `/`, ĝi estas relativa al dosierindiko en la medivariablo `INKL`.

Inkluzivigoj estas instrukcioj, sed estas nur utila por la organizo de la kodo kaj estas anstataŭigi per la kodo de la inkluzivenda dosiero antaŭe la sintaksa analizo.

2.4.3 Traktado de la fontkodo

Por komencigi la traktado de la fontkodo, fontdosiero devas esti provizata. Poste, la kodo estas traktata kiel ĉi tiu:

- La inkluzivigoj estas etendata (se inkluziva instrukcio estas malvalida, la traktado haltas);
- La komentoj estas forigiata, la konstantoj rekonigata, kaj la kodo estas transformata en ĵetonoj;

- La analizo de ĵetonoj startas. Se ie, la sintakso estas malvalida, la traktado haltas. Difinoj kaj deklaroj estas rekonigata kaj la valideco de la medioj kaj sekvencoj de instrukcioj estas regata;
- La traktado finas de post ĉiu ĵetonoj estas analizata, kaj nur en tiu kaso, programaro povas esti kreata.

2.4.4 Enirejo

La programaro startas kun la alvoko de la funkcio de enirejo, kiu havas nomon `enirejo` kaj malplenan enigon kaj eligon (enirejo kun argumentoj venos en poste versioj de Progvo).

Se la programaro ne havas enirejon, ĝi ne povas esti rekte plenumata (en poste versio de Progvo, ĝi povus esti uzata kiel biblioteko).

2.4.5 Nedifinita konduto

Kelkaj situacioj, indikota en la dokumentado, kaŭzas *nedifinita konduto*: la programaro eniras en stato, kiu ne estas difinata en la dokumentado. La realigoj de Progvo povas elekti, kiel trakti la plenumon en nedifinitaj kondutoj.

3. Baza biblioteko

3.1 Prezento

La baza biblioteko devus proponi la kapablojn, kiu estas sufiĉe por kodi ĉiujn programarojn en Progvo, en praktika fasono. Tamen, Progvo estas nova kaj ankoraŭ malkompleta, multe de kapabloj venos en poste versioj.

Ĉiu plenaj realigoj de Progvo devas proponi la funkciojn en tiu ĉapitro. Programaroj uzanta la bazan bibliotekan devas havi la konduton indikata en la dokumentado.

Simplaj realigoj de Progvo povas ne proponi ĉiujn funciojn. Ekzemple, enkorpigitaj sistemoj ne bozonas de realigoj de arbitraj grandaj nombroj kaj povus anstataŭe proponi nur indiĝenajn nombrojn.

Atributoj de eksteraj klasoj estas difinata en la dokumentado, sed la realigoj de klasoj ne bezonas strikte realigi ilin. Tamen, la klasoj devas havi kongruan konduton kun strikta realigo.

Ĉiu klaso kaj funkcio de la baza biblioteko estas en la nomspaco `Baz` kaj povas estis ekstera. La realigo de Progvo devas provizi la inkluzivendan dosieron `/Bazo/Bazo.Pvo`, kiu enhavas la difinojn kaj deklarojn de la klasoj kaj funkcioj de la baza biblioteko.

3.2 Bazaj klasoj

3.2.1 Buleo

La ekstera klaso `Buleo` permesas de uzi buleajn valorojn.

En la fontkodo, oni povas krei konstantajn buleojn kun `VERA` aŭ `MALVERA`.

Atributoj

La klaso ne havas atributon, sed objektoj havas du eblajn statojn (valorojn), vera kaj malvera.

Metodoj

La klaso ne havas metodon.

Rilataj funkcioj

Komparo `/= : (KONSTANTO Buleo' a, KONSTANTO Buleo' b) → (Buleo)`, vera se $a = b$, malvera alie.

Ne `/¬ : (KONSTANTO Buleo' a) → (Buleo b)`, kun $b = \neg a$.

Aŭ `/∨ : (KONSTANTO Buleo' a, KONSTANTO Buleo' b) → (Buleo c)`, kun $c = a \vee b$.

Kaj `/∧ : (KONSTANTO Buleo a, KONSTANTO Buleo' b) → (Buleo c)`, kun $c = a \wedge b$.

3.2.2 Bitsekvenco

La ekstera klaso `Bitsekvenco` permesas de konservi kaj manipuli sekvencojn de bitoj.

Bitsekvenco estas sekvencoj de $n \in \mathbb{N}$ bitoj (0 aŭ 1). En la fontkodo, oni povas krei konstantan bitsekvencon kun la sekvantaj sintaksoj:

- Rekte: `[`B`]`, kun $B \in \{0;1\}^*$. Ekzemple, `[01011]` prezentas sekvencon de 5 bitoj, kies unua estas 0 kaj lasta estas 1;
- Kun nombro: `#`B`[`N`]`, kun $B \in \{1; 2; ...; 16\}$ bazo kaj $N \in \mathbb{N}$ nombro, skribata en la ĝusta bazo (la ciferoj estas `0123456789ABCDEF`). La traktado de la kodo konvertas la nombron en duuma baza, kiel pezfina bitsekvenco, sen antaŭira nulo. Ekzemple, `[010111]`, `#2[111010]` kaj `#10[58]` estas la sama bitsekvenco. Se $B = 10$, oni ne bezonas precizi ĝin: `#[58]` estas sama ol `#10[58]`;
- Kun Ekia teksto: `"[`T`]`, kun T Ekio ĉeno, kun kelkaj da speciala reguloj: `]`, §, \ kaj signoj $s \notin \Sigma$ ne povas esti rekte uzata. Ili devas uzi la eskapilo \, kiu devas esti sekvata kun ilia Ekio numero en *du-cifera* deksesuma nombro (aliaj signoj povas ankaŭ uzi la eskapilon). Ĉiu signo kongruas kun 8 bitoj. Ekzemple, `"[abc]` estas la sama `Bitsekvenco` ol `[000100000010010001010]`. `"[\5B★/]` estas la sama ol `[010110110111110101011010]` (kaj prezentas la ĉenon `\★/`).

Atributoj

La klaso ne havas atributon, sed la objektoj havas staton, kongruanta al la bitsekvenco ĝi prezentas.

Metodoj

Konstruiloj

Kun buleo `KONSTRUILO (KONSTANTO Buleo')`
La stato de la kreata objekto kongruas la valoron de la eniga buleo.

Kunmeto kun alia `Bitsekvenco`: `kunmeti : (KONSTANTO Bitsekvenco') → ()`

Longo `longo : () → (Ind/N)`

Atingi la $(i+1)$-a biton: `get : (KONSTANTO Ind/N' i) → (Buleo)`
Se i estas pli granda aŭ egala ol la longo de la bitsekvenco, la alvoko de tiu funkcio kaŭzas nedifinitan konduton.

Modifi la $(i+1)$-a biton: `set : (KONSTANTO Buleo' b, KONSTANTO Ind/N' i) → ()`
Se i estas pli granda aŭ egala ol la longo de la bitsekvenco, la alvoko de tiu funkcio kaŭzas nedifinitan konduton.

Rilataj funkcioj

Komparo `/= : (KONSTANTO Bitsekvenco' a, KONSTANTO Bitsekvenco' b) → (Buleo)`, vera se a kaj b prezentas saman bitsekvencon, malvera alie.

Laŭbita Ne `/¬ : (KONSTANTO Bitsekvenco' a) → (Bitsekvenco b)`

Laŭbita Aŭ `/∨ : (KONSTANTO Bitsekvenco' a, KONSTANTO Bitsekvenco' b) → (Bitsekvenco c)`

Se la bitsekvencoj ne havas samajn longojn, la alvoko de tiu funkcio kaŭzas nedifinitan konduton.

Laŭbita Kaj `/∧ : (KONSTANTO Bitsekvenco a, KONSTANTO Bitsekvenco' b) → (Bitsekvenco c)`

Se la bitsekvencoj ne havas samajn longojn, la alvoko de tiu funkcio kaŭzas nedifinitan konduton.

Kunmeto `kunmeti : (Bitsekvenco, Bitsekvenco) → (Bitsekvenco)`

3.2.3 Ekio

La ekstera klaso `Ekio` permesas de konservi kaj manipuli Ekiajn ĉenojn.

Atributoj

La klaso havas unu atributon, de klaso `Bitsekvenco`, enmemoriganta la bitsekvencon kiu prezentas la Ekian ĉenon.

Metodoj

Konstruiloj

Kun bitsekvenco `KONSTRUILO (KONSTANTO Bitsekvenco')`

La prezentata Ekia ĉeno de la kreata objekto kongruas la bitsekvenco (unu bitoko estas unu signo). Se ĝia longo ne estas oblo de 8, la bitoj poste la lasta bitoko estas ignorinta.

Kunmeto kun alia `Ekio`: `kunmeti : (KONSTANTO Ekio') → ()`

Longo `longo : () → (Ind/N)` (nombro de signoj)

Atingi la $(i+1)$-a signon: `get : (KONSTANTO Ind/N' i) → (Bitsekvenco)`

La bitsekvenco estas la bitoko prezanta la signon.

Se i estas pli granda aŭ egala ol la longo de la ĉeno, la alvoko de tiu funkcio kaŭzas nedifinitan konduton.

Modifi la $(i+1)$-a signon: `set : (KONSTANTO Bitsekvenco' b, KONSTANTO Ind/N' i) → ()`

La bitsekvenco estas la bitoko prezanta la signon.

Se i estas pli granda aŭ egala ol la longo de la ĉeno, aŭ se la bitsekvenco ne estas bitoko, la alvoko de tiu funkcio kaŭzas nedifinitan konduton.

Rilataj funkcioj

Komparo `/= : (KONSTANTO Ekio' a, KONSTANTO Ekio' b) → (Buleo)`, vera se a kaj b prezentas saman ĉenon, malvera alie.

Kunmeto `kunmeti : (KONSTANTO Ekio', KONSTANTO Ekio') → (Ekio)`

3.2.4 N

Tiu klaso prezentas la matematikan aron $\mathbb{N}$. Nombroj havas arbitrajn grandojn.

En la fontkodo, oni povas krei konstantan objekton de klaso $\mathbb{N}$ kun vorto de la aro

$$K_N = (\{0\} \cup \{1;2;...;9\}\{0;1;...;9\}^*)\{N\} \subset K$$

$K_N{}'$ estas vortoj sen sufikso `N`: $\{0\} \cup \{1;2;...;9\}\{0;1;...;9\}^*$

Nombro povas esti malfinia (∞) aŭ malvalida (nenombro, la Ekia signo 251 prezentas ĝin).

Atributoj

La klaso havas unu atributon de klaso `Bitsekvenco`, enmemoriganta la bitsekvencon kiu prezentas la nombron. Ĝi ankaŭ havas du `Buleo`jn, veraj se la nombro estas respektive malvalida aŭ malfinia.

Metodoj

Konstruiloj

Kun bitsekvenco `KONSTRUILO (KONSTANTO Bitsekvenco')`

La prezentata nombro de la kreata objekto kongruas la bitsekvencon (la bitoj estas analizita kiel duuma nombro). Se ĝia longo estas 0, la nombro estas malvalida.

Kun Ekia ĉeno `KONSTRUILO (KONSTANTO Ekio')`

La prezentata nombro de la kreata objekto kongruas la ĉenon. Se la ĉeno ne prezentas validan naturan nombron, la nombro estas malvalida.

Alkrementado `alk : () → ()`

Por nombro `n` de klaso `N`, ĝia valoro poste `n.alk()` estos sama ol la valoro poste `n ← +(n, 1N)`.

Dekrementado `dek : () → ()`

Por nombro `n` de klaso `N`, ĝia valoro poste `n.dek()` estos sama ol la valoro poste `n ← -(n, 1N)`.

Rilataj funkcioj

Komparoj `/= : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a = b$, malvera alie.

`/< : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a < b$, malvera alie.

`/<= : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a \leq b$, malvera alie.

`/> : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a > b$, malvera alie.

`/>= : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a \geq b$, malvera alie.

Adicio `/+ : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = a + b$.

Subtraho `/- : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = a - b$.

Obligo `/× : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = a \times b$.

Divido `/÷ : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = \lfloor \frac{a}{b} \rfloor$.

Resto de divido `/mod : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = a - b \left\lfloor \frac{a}{b} \right\rfloor$.

Apartaj kasoj

- Se a aŭ b estas nenombroj, la komparoj estas ĉiam malvera kaj la aritmetikaj operacioj eligas nenombron;
- Komparoj estos ankaŭ malvera se ambaŭ nombroj estas malfinia. Alie, se nur unu nombro estas malfinia, la rezulto estas intuicia (ekzemple, $\infty > a$ estas vera por ĉiu objekto a de klaso `N`);
- Adicio: $a + \infty = \infty$, $\infty + b = \infty$
- Subtraho: $a - b$ estas nenombro se $a < b$ aŭ $b = \infty$. Alie, $\infty - b = \infty$;
- Obligo: $0 \times \infty$ kaj $\infty \times 0$ estas nenombroj. Alie, se a aŭ b estas malfinia, c estas malfinia;
- Divido: $\infty \div \infty$ kaj $0 \div 0$ estas nenombroj. Alie, $\infty \div b = \infty$ kaj $a \div 0 = \infty$;
- Resto de divido: eligo estas nenombro se $a = \infty$ aŭ $b = 0$. $a \mod \infty = a$.

3.2.5 Z

Tiu klaso prezentas la matematikan aron $\mathbb{Z}$. Nombroj havas arbitrajn grandojn.

En la fontkodo, oni povas krei konstantan objekton de klaso `Z` kun vorto de la aro

$$K_Z = (\{0\} \cup \{\epsilon, -\}\{1;2;...;9\}\{0;1;...;9\}^*)\{Z\} \subset K$$

$K_Z{}'$ estas vortoj sen sufikso `Z`: $\{0\} \cup \{\epsilon, -\}\{1;2;...;9\}\{0;1;...;9\}^*$.

Nombro povas esti malfinia ($\pm\infty$) aŭ malvalida (nenombro, la Ekia signo 251 prezentas ĝin).

Atributoj

La klaso havas unu atributon de klaso `Bitsekvenco`, enmemoriganta la bitsekvencon kiu prezentas la nombron. Ĝi ankaŭ havas tri `Buleo`jn, veraj se la nombro estas respektive negativa, malvalida aŭ malfinia. Nombro 0 povas nur esti positiva.

Metodoj

Konstruiloj

Kun bitsekvenco `KONSTRUILO (KONSTANTO Bitsekvenco')`

La prezentata nombro de la kreata objekto kongruas la bitsekvenco (la unua bito estas la signo, la aliaj estas analizita kiel duuma nombro). Se ĝia longo estas 0 aŭ 1, la nombro estas malvalida.

Kun Ekia ĉeno `KONSTRUILO (KONSTANTO Ekio')`

La prezentata nombro de la kreata objekto kongruas la ĉeno. Se la ĉeno ne prezentas validan entjeron, la nombro estas malvalida.

Kun natura nombro `KONSTRUILO (KONSTANTO N')`

La prezentata nombro de la kreata objekto kongruas la nombron en enigo.

Alkrementado `alk : () → ()`

Por nombro `n` de klaso `Z`, ĝia valoro poste `n.alk()` estos sama ol la valoro poste `n ← +(n, 1Z)`.

Dekrementado `dek : () → ()`

Por nombro `n` de klaso `Z`, ĝia valoro poste `n.dek()` estos sama ol la valoro poste `n ← -(n, 1Z)`.

Rilataj funkcioj

Komparoj `/= : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a = b$, malvera alie.

`/< : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a < b$, malvera alie.

`/<= : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a \leq b$, malvera alie.

`/> : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a > b$, malvera alie.

`/>= : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a \geq b$, malvera alie.

Kontraŭegalo `/- : (KONSTANTO Z' a) → (Z b)`, kun $b = -a$.

Adicio `/+ : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = a + b$.

Subtraho `/- : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = a - b$.

Obligo `/× : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = a \times b$.

Divido `/÷ : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = \left\lfloor \frac{a}{b} \right\rfloor$.

Resto de divido `/mod : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = a - b \left\lfloor \frac{a}{b} \right\rfloor$.

Apartaj kasoj

- Se a aŭ b estas nenombroj, la komparoj estas ĉiam malvera kaj la aritmetikaj operacioj eligas nenombron;
- Komparoj estas ankaŭ malvera se ambaŭ nombroj estas malfinia kun sama signo. Alie, la rezulto estas intuicia (ekzemple, $+\infty > a$ estas vera por ĉiu objekto a de klaso `Z`);
- Adicio kaj subtraho ($a - b$ estas $a + (-b)$): $-\infty + \infty$ kaj $\infty + (-\infty)$ estas nenombroj. Alie, $a + (\pm\infty) = \pm\infty$, $\pm\infty + b = \pm\infty$;
- Obligo: $0 \times (\pm\infty)$ kaj $\pm\infty \times 0$ estas nenombroj. Alie, se a aŭ b estas malfinia, c estas malfinia, kun la propa signo;
- Divido: $\pm\infty \div (\pm\infty)$ kaj $a \div 0$ estas nenombroj. Alie, se a estas malfinia, c estos malfinia, kun la propa signo, kaj $a \div \pm\infty = 0$;
- Resto de divido: eligo estas nenombro se $a = \pm\infty$ aŭ $b = 0$. $a \mod \pm\infty = a$.

3.2.6 Q

Tiu klaso prezentas la matematikan aron $\mathbb{Q}$. Nombroj havas arbitrajn grandojn.

En la fontkodo, oni povas krei konstantan objekton de klaso `Q` kun vorto de la aro

$$K_Q = (\{0\} \cup K_Z{}' \cup K_Z{}'\{/\}K_N{}' \cup K_Z{}'\{/\}\{(\}K_Z{}'\{)\})\{Q\} \subset K_{nombroj}$$

$K_Q{}'$ estas vortoj sen sufikso `Q`: $\{0\} \cup K_Z{}' \cup K_Z{}'\{/\}K_N{}' \cup K_Z{}'\{/\}\{(\}K_Z{}'\{)\}$.

Nombro povas esti malfinia ($\pm\infty$) aŭ malvalida (nenombro, la Ekia signo 251 prezentas ĝin).

Atributoj

La klaso havas du atributojn de klaso `Z`, enmemoriganta la numeratoron kaj la denominatoron. La negativa, malvalida kaj malfinia ecoj de la nombro estas determinita kun la du nombroj:

- La nombro estas malvalida se numeratoron aŭ la denominatoron estas malvalida, se ambaŭ nombroj estas malfiniaj, aŭ se la denominatoro estas nulo;
- Alie, la nombro estas malfinia se la numeratoro estas malfinia;
- La signo estas positiva se ambaŭ nombroj havas saman signon, negativa alie.

Metodoj

Konstruiloj

Kun Ekia ĉeno `KONSTRUILO (KONSTANTO Ekio')`

La prezentata nombro de la kreata objekto kongruas la ĉenon. Se la ĉeno ne prezentas validan naturan nombron, la nombro estas malvalida.

Kun entjero `KONSTRUILO (KONSTANTO N')`
`KONSTRUILO (KONSTANTO Z')`

La prezentata nombro de la kreata objekto kongruas la nombron en enigo.

Kun numeratoro kaj denominatoro `KONSTRUILO (KONSTANTO Z' a, KONSTANTO Z' b)`

La prezentata nombro de la kreata objekto estas $\frac{a}{b}$.

Alkrementado `alk : () → ()`

Por nombro `n` de klaso `Q`, ĝia valoro poste `n.alk()` estos sama ol la valoro poste `n ← +(n, 1Q)`.

Dekrementado `dek : () → ()`

Por nombro `n` de klaso `Q`, ĝia valoro poste `n.dek()` estos sama ol la valoro poste `n ← -(n, 1Q)`.

Rilataj funkcioj

Komparoj `/= : (KONSTANTO Q' a, KONSTANTO Q' b) → (Buleo)`, vera se $a = b$, malvera alie.
`/< : (KONSTANTO Q' a, KONSTANTO Q' b) → (Buleo)`, vera se $a < b$, malvera alie.
`/<= : (KONSTANTO Q' a, KONSTANTO Q' b) → (Buleo)`, vera se $a \leq b$, malvera alie.
`/> : (KONSTANTO Q' a, KONSTANTO Q' b) → (Buleo)`, vera se $a > b$, malvera alie.
`/>= : (KONSTANTO Q' a, KONSTANTO Q' b) → (Buleo)`, vera se $a \geq b$, malvera alie.

Kontraŭegalo `/- : (KONSTANTO Q' a) → (Q b)`, kun $b = -a$.

Adicio `/+ : (KONSTANTO Q' a, KONSTANTO Q' b) → (Q c)`, kun $c = a + b$.

Subtraho `/- : (KONSTANTO Q' a, KONSTANTO Q' b) → (Q c)`, kun $c = a - b$.

Obligo `/×` `: (KONSTANTO Q' a, KONSTANTO Q' b) → (Q c)`, kun $c = ab$.

Divido `/÷ : (KONSTANTO Q' a, KONSTANTO Q' b) → (Q c)`, kun $c = \frac{a}{b}$.

Apartaj kasoj Ili estas sama ol ke la klaso `Z`.

3.3 Bazaj funkcioj

3.3.1 Funkcioj de eneligo

Realigoj de Progvo devas proponi bazajn kapablojn de eneligo: terminalo, por baza afiŝado kaj enigo de teksto, aŭ manipuloj de dosieroj. La sekvantaj funkcioj permesas de uzi tiojn kapablojn.

Enigo en terminalo Tioj funkcioj permesas de legi kaj interpreti la enigon de Ekia ĉeno de la uzanto, en terminalo.

De Buleo `enigi : (Buleo') → ()`
La buleo estas vera se la enigo estas `VERA` aŭ `1`, malvera alie.

De Bitsekvenco `enigi : (Bitsekvenco') → ()`
La enigo devas enhavi nur `0` kaj `1`. Alie, la sekvenco estas malplena.

De Ekio ĉeno `enigi : (Ekio') → ()`

De nombro `enigi : (N') → ()`
`enigi : (Z') → ()`
`enigi : (Q') → ()`
La enigo devas esti valida. Alie, la nombro estos nenombro.

Afiŝado

De Buleo `printi : (KONSTANTO Buleo') → ()`
Tio printas `VERA` aŭ `MALVERA`.

De Bitsekvenco `printi : (KONSTANTO Bitsekvenco') → ()`
Tio printas la bitojn de la `Bitsekvenco` kun la signoj `0` kaj `1`.

De Ekio ĉeno `printi : (KONSTANTO Ekio') → ()`

De nombro `printi : (KONSTANTO N') → ()`
`printi : (KONSTANTO Z') → ()`
`printi : (KONSTANTO Q') → ()`

Dosieroj

Ŝargado `ŝargi : (KONSTANTO Ekio' di) → (Bitsekvenco')`
`di` estas la dosierindiko de la dosiero. La `Bitsekvenco` estas ĝia datumo.

Skribado `skribi : (Bitsekvenco' datumo, KONSTANTO Ekio' di) → ()`

4. Indiĝenaj kapabloj

4.1 Prezento

Realigoj de Progvo devus proponi la kapablojn en tiu ĉapitro, se ebla. Iliaj celoj estas permesi malaltnivelajn kapablojn aŭ rendimentajn komputadojn uzanta indiĝenajn kapabloj.

Ĉiu klaso kaj funkcio de la indiĝena biblioteko estas en la nomspaco `Ind` kaj estas ekstera. La realigo de Progvo devas provizi la inkluzivenda dosiero `/Bazo/Indi.Pvo`, kiu enhavas la deklarojn de la klasoj kaj funkcioj de la indiĝena biblioteko.

4.2 Indiĝenaj klasoj

4.2.1 Adreso

La ekstera klaso `Adreso` permesas de konservi kaj manipuli memoradresojn. En aliaj vortoj, la objektoj de klaso Adreso estas adresmontriloj. Ĝi permesas tre malaltnivelajn operaciojn.

En la kodo, oni povas krei konstantan objekton kun vorto de $K_N{}'$.

Atributoj

La klaso havas unu atributon de klaso `Bitsekvenco`, enmemoriganta la bitsekvencon kiu prezentas nombron, kiu estas la memoradreso. Ĝia longo estas konstanta kaj devas kongrui la longon de memoradresoj.

Metodoj

Se la programaro estas plenumita en operaciumo, legi kaj skribi al neasignata memorareoj kaŭzas nedifinitan konduton.

Asignado `asigni : (KONSTANTO N' n) → (Adreso)`

Asignas koneksan memorareon, sufice granda por n bitoj. La enigo estas la adreso de la unua bito.

Lego `get : (KONSTANTO N' n) → (Bitsekvenco)`

La bitsekvenco estas la n bitoj al kaj poste la adreso.

Skribo `set : (Bitsekvenco) → ()`

Skribas la bitsekvencon al la adreso (unua bito) kaj la sekvantaj.

4.2.2 N

Tiu klaso prezentas la matematikan aron $\mathbb{N}$. La nombroj devas esti indigênajn entierojn de la sistemo (ekzemple nombroj de 64 bitoj en x64 komputilojn) sed dependas ankaŭ en la realigo de Progvo.

En la fontkodo, oni povas krei konstantan objekton de klaso `N` kun vorto de la aro

$$K_n = (\{0\} \cup \{1;2;...;9\}\{0;1;...;9\}^*)\{n\} \subset K$$

Atributoj

La klaso havas unu atributon de klaso `Bitsekvenco`, enmemoriganta la bitsekvencon kiu prezentas la nombron (ĝia longo estas konstanta).

Metodoj

Konstruiloj

Kun bitsekvenco `KONSTRUILO (KONSTANTO Bitsekvenco')`

La bitsekvenco anstataŭigas tion de la objekto. Ilia longoj devas esti sama, alie nedifinita konduto estas kaŭzata.

Kun Ekia ĉeno `KONSTRUILO (KONSTANTO Ekio')`

La prezentata nombro de la kreata objekto kongruas la ĉenon. Se la ĉeno ne prezentas validan naturan nombron aŭ estas tro granda, nedifinita konduto estas kaŭzata.

Alkrementado `alk : () → ()`

Por nombro `n` de klaso `N`, ĝia valoro poste `n.alk()` estos sama ol la valoro poste `n ← +(n, 1n)`.

Dekrementado `dek : () → ()`

Por nombro `n` de klaso `N`, ĝia valoro poste `n.dek()` estos sama ol la valoro poste `n ← -(n, 1n)`.

Rilataj funkcioj

Komparoj `/= : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a = b$, malvera alie.
`/< : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a < b$, malvera alie.
`/<= : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a \leq b$, malvera alie.
`/> : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a > b$, malvera alie.
`/>= : (KONSTANTO N' a, KONSTANTO N' b) → (Buleo)`, vera se $a \geq b$, malvera alie.

Adicio `/+ : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = a + b$.

Subtraho `/- : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = a - b$.

Obligo `/× : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = a \times b$.

Divido `/÷ : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = \left\lfloor \frac{a}{b} \right\rfloor$.

Resto de divido `/mod : (KONSTANTO N' a, KONSTANTO N' b) → (N c)`, kun $c = a - b \left\lfloor \frac{a}{b} \right\rfloor$.

Apartaj kasoj La komparoj kaj operacioj estas indiĝena komputata kun la maŝino/ĉefprocesoro. Kasoj kiel trooj aŭ divido per nulo kaŭzas nedifinitan konduton.

4.2.3 Z

Tiu klaso prezentas la matematikan aron $\mathbb{Z}$. La nombroj devas esti indigênajn entierojn de la sistemo kun sama longo ol `N`.

En la fontkodo, oni povas krei konstantan objekton de klaso `Z` kun vorto de la aro

$$K_z = (\{0\} \cup \{\epsilon, -\}\{1;2;...;9\}\{0;1;...;9\}^*)\{z\} \subset K$$

Atributoj

La klaso havas unu atributon de klaso `Bitsekvenco`, enmemoriganta la bitsekvencon kiu prezentas la nombron (ĝia longo estas konstanta).

Metodoj

Konstruiloj

Kun bitsekvenco `KONSTRUILO (KONSTANTO Bitsekvenco')`

La bitsekvenco anstataŭigas tion de la objekto. Ilia longoj devas esti sama, alie nedifinita konduto estas kaŭzata.

Kun Ekia ĉeno `KONSTRUILO (KONSTANTO Ekio')`

La prezentata nombro de la kreata objekto kongruas la ĉenon. Se la ĉeno ne prezentas validan entjeron aŭ estas tro granda, nedifinita konduto estas kaŭzata.

Kun natura nombro `KONSTRUILO (KONSTANTO N')`

La prezentata nombro de la kreata objekto kongruas la nombron en enigo.

Alkrementado `alk : () → ()`

Por nombro `n` de klaso `Z`, ĝia valoro poste `n.alk()` estos sama ol la valoro poste `n ← +(n, 1z)`.

Dekrementado `dek : () → ()`

Por nombro `n` de klaso `Z`, ĝia valoro poste `n.dek()` estos sama ol la valoro poste `n ← -(n, 1z)`.

Rilataj funkcioj

Komparoj `/= : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a = b$, malvera alie.

`/< : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a < b$, malvera alie.

`/<= : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a \leq b$, malvera alie.

`/> : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a > b$, malvera alie.

`/>= : (KONSTANTO Z' a, KONSTANTO Z' b) → (Buleo)`, vera se $a \geq b$, malvera alie.

Kontraŭegalo `/- : (KONSTANTO Z' a) → (Z b)`, kun $b = -a$.

Adicio `/+ : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = a + b$.

Subtraho `/- : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = a - b$.

Obligo `/×  : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = a \times b$.

Divido `/÷ : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = \left\lfloor \frac{a}{b} \right\rfloor$.

Resto de divido `/mod : (KONSTANTO Z' a, KONSTANTO Z' b) → (Z c)`, kun $c = a - b \left\lfloor \frac{a}{b} \right\rfloor$.

Apartaj kasoj La komparoj kaj operacioj estas indiĝena komputita kun la maŝino/ĉefprocesoro. Kasoj kiel trooj aŭ divido per nulo kaŭzas nedifinitan konduton.

4.2.4 R

Tiu klaso prezentas la matematikan aron $\mathbb{R}$. La nombroj devas esti indigênaj reeloj de la sistemo (ekzemple *IEEE 754 Extended Precision* nombroj en x64 komputilojn) sed dependas en la realigo de Progvo.

En la fontkodo, oni povas krei konstantan objekton de klaso `R` kun vorto de la aro

$$K_r = (\{0\} \cup K_Z{}' \cup K_Z{}'\{.\}K_N{}')\{r\} \subset K$$

Nombro povas esti malfinio ($\pm\infty$) aŭ malvalida (nenombro, la Ekia signo 251 prezentas ĝin).

Atributoj

La klaso havas unu atributon de klaso `Bitsekvenco`, enmemoriganta la bitsekvencon kiu prezentas la nombron (ĝia longo estas konstanta).

Metodoj

Konstruiloj

Kun bitsekvenco `KONSTRUILO (KONSTANTO Bitsekvenco')`

La bitsekvenco anstataŭigas tion de la objekto. Ilia longoj devas esti sama, alie nedifinita konduto estas kaŭzata.

Kun Ekia ĉeno `KONSTRUILO (KONSTANTO Ekio')`

La prezentata nombro de la kreata objekto kongruas la ĉenon. Se la ĉeno ne prezentas validan reelon aŭ estas tro granda, nedifinita konduto estas kaŭzata.

Kun entjero `KONSTRUILO (KONSTANTO N')`

`KONSTRUILO (KONSTANTO Z')`

La prezentata nombro de la kreata objekto kongruas la nombron en enigo.

Alkrementado `alk : () → ()`

Por nombro `n` de klaso `R`, ĝia valoro poste `n.alk()` estos sama ol la valoro poste `n ← +(n, 1r)`.

Dekrementado `dek : () → ()`

Por nombro `n` de klaso `R`, ĝia valoro poste `n.dek()` estos sama ol la valoro poste `n ← -(n, 1r)`.

Rilataj funkcioj

Komparoj `/= : (KONSTANTO R' a, KONSTANTO R' b) → (Buleo)`, vera se $a = b$, malvera alie.
`/< : (KONSTANTO R' a, KONSTANTO R' b) → (Buleo)`, vera se $a < b$, malvera alie.
`/<= : (KONSTANTO R' a, KONSTANTO R' b) → (Buleo)`, vera se $a \leq b$, malvera alie.
`/> : (KONSTANTO R' a, KONSTANTO R' b) → (Buleo)`, vera se $a > b$, malvera alie.
`/>= : (KONSTANTO R' a, KONSTANTO R' b) → (Buleo)`, vera se $a \geq b$, malvera alie.

Kontraŭegalo `/- : (KONSTANTO R' a) → (R b)`, kun $b = -a$.

Adicio `/+ : (KONSTANTO R' a, KONSTANTO R' b) → (R c)`, kun $c = a + b$.

Subtraho `/- : (KONSTANTO R' a, KONSTANTO R' b) → (R c)`, kun $c = a - b$.

Obligo `/×` `: (KONSTANTO R' a, KONSTANTO R' b) → (R c)`, kun $c = ab$.

Divido `/÷` `: (KONSTANTO R' a, KONSTANTO R' b) → (R c)`, kun $c = \frac{a}{b}$.

Apartaj kasoj Ili dependas en la maŝino kaj la realigo de Progvo.

4.3 Indiĝenaj funkcioj

4.3.1 Funkcioj de eneligo

Enigo en terminalo

De nombro `enigi : (N') → ()`
`enigi : (Z') → ()`
`enigi : (R') → ()`
La enigo devas esti valida. Alie, la nombro estas nenombro.

Afiŝado

De adreso `printi : (KONSTANTO Adreso') → ()`
Tio printas la numero de la memoradreso.

De nombro `printi : (KONSTANTO N') → ()`
`printi : (KONSTANTO Z') → ()`
`printi : (KONSTANTO R') → ()`

5. Grafika biblioteko

Progvo povas proponi bazajn grafikajn kapablojn. La klasoj kaj funkcioj de la grafika biblioteko estas en la nomspaco `Baz` kaj povas esti ekstera. La realigo de Progvo provizus la inkluzivenda dosiero `/Bazo/Grafiko.Pvo`, kiu enhavas la deklarojn kaj difinojn de la klasoj kaj funkcioj de la grafika biblioteko.

5.1 Klasoj

La klasoj estos plibonigata kaj kompletigata en poste versioj de Progvo.

5.1.1 Kanvaso

Tiu klaso prezentas kanvasojn. Kanvaso estas dudimensia tabelo de bilderoj. La difino ne estas pli preciza, ĉar tio permesas realigojn de Progvo de adapti al la sistemo (kanvaso povus esti fenestroj en operaciumo kun fenestroj, aŭ la plena ekrano en enkorpigita sistemo kun tre simpla grafika interfaco).

Atributoj

La klaso havas unu atributon de klaso `Bitsekvenco`, enmemoriganta la datumo de bilderoj (ĝia longo estas konstanta).

Metodoj

Konstruiloj

Kun dimensioj `KONSTRUILO (KONSTANTO Ind/N' l, KONSTANTO Ind/N' a)`

Kreas kanvason de larĝo l kaj alto a (bilderoj).

Validaj pozicioj estas horizontale 0-$(l-1)$ kaj vertikale 0-$(a-1)$.

Ŝanĝi bilderon `setBilderon : (KONSTANTO Ind/N' r, KONSTANTO Ind/N' v, KONSTANTO Ind/N' b, KONSTANTO Ind/N' x, KONSTANTO Ind/N' y) → ()`

Ŝanĝas la koloron de la bildero kun pozicio (x, y). Se la pozicio estas malvalida, nedifinita konduto estas kaŭzata.

Akiri bilderon `getBilderon : (KONSTANTO Ind/N' x, KONSTANTO Ind/N' y) → (Ind/N r, Ind/N v, Ind/N b)`

Akiras la koloron de la bildero kun pozicio (x, y). Se la pozicio estas malvalida, nedifinita konduto estas kaŭzata.

5.2 Fenestraĵoj

Venos en poste versio de Progvo.