Pracovní list 6: Abstraktní datové typy

Co už máme znát

- Pojem datového typu
- Konkrétní a abstraktní typ
- Popis hodnot datového typu
- Diagram signatury
- Sémantika operací
- Dynamické datové struktury
- Programové moduly

Kontrolní otázky

6.1 Čím je definován datový typ?

6.2 Co je abstraktní a co je konkrétní datový typ?

6.3 Jakými formalismy lze popisovat hodnoty datového typu?

6.4 Co určuje a jak je definován diagram signatury?

6.5 Co řeší sémantické definice a jak je lze zapisovat?

6.6 Jak lze implementovat abstraktní datový typ?

6.7 Jak se implementují programové moduly?

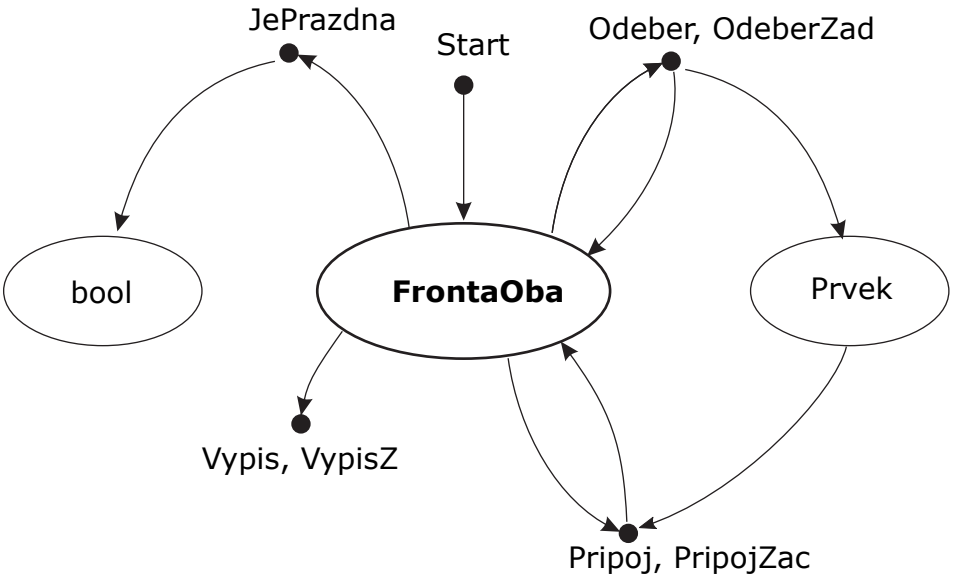
6.8 Který soubor vznikne přepisem diagramu signatury?

Příprava na cvičení

Ve cvičení budeme potřebovat překladač jazyka C++, editor pro přípravu zdrojových textů a vybavení příkazového řádku, papír, dobře ořezanou měkkou tužku a gumu. Pro jednotlivé úlohy jsou k dispozici soubory s daty, případně výsledné soubory v adresáři `/home/rybicka/vyuka/progt/cecko/cviceni/cv06` na serveru akela. Konkrétní jména těchto souborů jsou uvedena u jednotlivých úloh.

Řešené příklady

Příklad 6.1 Uvažujme abstraktní datový typ „Obousměrná fronta“ (identifikátor `FrontaOba`) a k němu následující diagram signatury:



Navrhněte hlavičkový soubor modulu implementující tento datový typ pomocí obousměrného dynamického lineárního seznamu. Předpokládejte, že prvkem fronty budou desetinná čísla.

Řešení:

```
#ifndef FRONTA2_H
#define FRONTA2_H
typedef double Prvek; //datová složka je desetinné číslo
struct Clen { //jeden člen seznamu
  Prvek Data;
  Clen *Nasled, *Predch; //dva ukazatele
};
typedef Clen * UkClen;
struct FrontaOba {
  UkClen Zac, Kon;  //fronta potřebuje ukazatel na začátek i konec
};

void Start(FrontaOba &F);
  //inicializace fronty
void Pripoj(FrontaOba &F, Prvek E);
  //přidání prvku na konec fronty
void PripojZac(FrontaOba &F, Prvek E);
  //přidání prvku na začátek fronty
Prvek Odeber(FrontaOba &F);
  //odebrání prvku z konce fronty
```

```
276 Prvek OdeberZad(FrontaOba &F);
277   //odebrání prvku ze začátku fronty
278 void Vypis(FrontaOba F);
279   //výpis obsahu fronty na standardní výstup
280 void VypisZ(FrontaOba F);
281   //výpis obsahu fronty zezadu na standardní výstup
282 bool JePrazdna(FrontaOba F);
283   //zjištění prázdnosti fronty
284 #endif
```

Příklad 6.2 Proveďte implementaci abstraktního typu Oboustranná fronta podle následujících sémantických definic:
Operace Start:

$$\Rightarrow \langle\rangle$$

Operace Pripoj:

$$x, \langle a_1, ..., a_n \rangle \Rightarrow \langle a_1, ..., a_n, x \rangle$$

Operace PripojZac:

$$x, \langle a_1, ..., a_n \rangle \Rightarrow \langle x, a_1, ..., a_n \rangle$$

Operace Odeber:

$$\begin{aligned} \langle\rangle &\Rightarrow \langle\rangle, 0 \\ \langle a_1, ..., a_{n-1}, a_n \rangle &\Rightarrow \langle a_1, ..., a_{n-1} \rangle, a_n \end{aligned}$$

Operace OdeberZad:

$$\begin{aligned} \langle\rangle &\Rightarrow \langle\rangle, 0 \\ \langle a_1, a_2, ..., a_n \rangle &\Rightarrow \langle a_2, ..., a_n \rangle, a_1 \end{aligned}$$

Operace Vypis:

$$\langle a_1, ..., a_{n-1}, a_n \rangle \Rightarrow a_1 \_ ... a_{n-1} \_ a_n \_$$

Operace VypisZ:

$$\langle a_1, ..., a_{n-1}, a_n \rangle \Rightarrow a_n \_ a_{n-1} \_ ... a_1 \_$$

Operace JePrazdna:

$$\begin{aligned} \langle\rangle &\Rightarrow \mathit{true} \\ \langle a_1, ... \rangle &\Rightarrow \mathit{false} \end{aligned}$$

Řešení: Vytvoříme implementační soubor s následujícím obsahem:

```
285 #include <iostream>
286 #include "fronta2.h"
287 using namespace std;
288
289 void Start(FrontaOba &F){
```

```
290   F.Zac=F.Kon=NULL;
291 }
292
293 void Pripoj(FrontaOba &F, Prvek E){
294   if (F.Zac==NULL){
295      F.Zac = new Clen;
296      F.Kon = F.Zac;
297      F.Kon->Predch = NULL;
298   } else {
299      F.Kon->Nasled = new Clen;
300      F.Kon->Nasled->Predch = F.Kon;
301      F.Kon = F.Kon->Nasled;
302   }
303   F.Kon->Data = E;
304 }
305
306 void PripojZac(FrontaOba &F, Prvek E){
307   UkClen Pom = new Clen;
308   Pom->Nasled = F.Zac;
309   if (Pom->Nasled != NULL) Pom->Nasled->Predch = Pom;
310   Pom->Predch = NULL;
311   Pom->Data = E;
312   F.Zac = Pom;
313   if (F.Kon==NULL) F.Kon=F.Zac;
314 }
315
316 Prvek Odeber(FrontaOba &F){
317   if (not JePrazdna(F)) {
318     return F.Zac->Data;
319     UkClen Pom = F.Zac;
320     F.Zac = F.Zac->Nasled;
321     F.Zac->Predch = NULL;
322     delete Pom;
323   } else return 0;
324 }
325
326 Prvek OdeberZad(FrontaOba &F){
327   if (not JePrazdna(F)) {
328     return F.Kon->Data;
329     UkClen Pom = F.Kon;
330     F.Kon = F.Kon->Predch;
331     F.Kon->Nasled = NULL;
332     delete Pom;
333   } else return 0;
334 }
```

```
335
336 void Vypis(FrontaOba F){
337   UkClen Pom = F.Zac;
338   while (Pom != NULL) {
339     cout << Pom->Data << ” ”;
340     Pom = Pom->Nasled;
341   }
342   cout << endl;
343 }
344
345 void VypisZ(FrontaOba F){
346   UkClen Pom = F.Kon;
347   while (Pom != NULL) {
348     cout << Pom->Data << ” ”;
349     Pom = Pom->Predch;
350   }
351   cout << endl;
352 }
353
354 bool JePrazdna(FrontaOba F){
355   return F.Zac == NULL;
356 }
```

Příklad 6.3 Modul implementovaný v předchozích úlohách vyzkoušejte v následující úloze: Na standardním vstupu je číslo N a za ním je N desetinných čísel. Vložte první polovinu do fronty zezadu, druhou polovinu čísel do fronty zepředu. Pak tuto frontu vypište zepředu i zezadu.

Řešení:

```
357 #include <iostream>
358 #include ”fronta2.h”
359
360 using namespace std;
361
362 int main(){
363   FrontaOba Moje;
364   Start(Moje);
365   int pocet;
366   double vstup;
367   cin >> pocet;
368   for (int i=0; i<pocet/2; i++){
369     cin >> vstup;
370     Pripoj(Moje, vstup);
371   }
372   while (cin>>vstup) PripojZac(Moje, vstup);
```

```
373     Vypis(Moje);
374     VypisZ(Moje);
375
376     return 0;
377 }
```

Příklady

Příklad 6.4 Nakreslete diagram signatury k úloze 5.8. Zapište sémantické definice všech operací tohoto diagramu. Jednotlivé operace budou mít následující identifikátory: `init` (inicializace seznamu), `vloz` (vložení prvku), `odeber` (odebrání prvku za aktivním prvkem), `je_akt` (zjištění, zda je seznam aktivní), `vypis_akt` (výpis obsahu aktivního prvku), `posun_akt` (přesun aktivity na následující prvek), `posun_akt_z` (přesun aktivity na začátek).

Příklad 6.5 Přepište diagram signatury z úlohy 6.4 do podoby hlavičkového souboru v jazyce C++.

Příklad 6.6 Upravte původní řešení příkladu 5.8 tak, aby odpovídalo navrženému diagramu signatury – vytvořte příslušný implementační soubor.

Příklad 6.7 Napište k vytvořenému hlavičkovému souboru druhý implementační soubor, který bude obsahovat implementaci operací z diagramu signatury pomocí pole. V hlavičkovém souboru proveďte potřebné změny (odvození datového typu).

Příklad 6.8 Vyměňte si implementační soubory se spolužákem a zkuste je přeložit a spustit s vlastním upraveným hlavičkovým souborem.

Co máme po cvičení umět

- Sestrojit diagram signatury k určitému datovému typu.
- Podle diagramu signatury vytvořit hlavičkový soubor programového modulu.
- Podle hlavičkového souboru implementovat operace s využitím datového typu pole.
- Podle hlavičkového souboru implementovat operace s využitím lineárního seznamu.

Kontrolní otázky

6.9 V čem spočívá výhoda návrhu abstraktního datového typu diagramem signatury?

6.10 Lze k jednomu hlavičkovému souboru mít více implementačních souborů?

6.11 K čemu je dobré mít alternativní implementace téhož abstraktního datového typu?