34. UNIJA

Unija je izvedeni tip podatka u C jeziku koji omogućava da se na istoj memorijskoj lokaciji memorišu podaci različitog tipa, naravno samo jedan u jednom datom trenutku, ali ne i više istovremeno.

Poput strukture kao izvedenog tipa i **uniju je potrebno prvo definisati da bi se mogla koristiti** u deklaracijama. **Sintaksa** je slična kao i kod strukture:

```
union naziv_unije // Definicija unije
{
      tip1 promenljiva1; // Definicija elemenata unije
      tip2 promenljiva2;
      …
      tipN promenljivaN;
} identifikator(i);
```

Deklaracija promenljivih tipa strukture – prilikom deklaracije promenljive tipa unije potrebno je navesti službenu reč union pa naziv unije i identifikator promenljive:

```
union naziv_unije promenljiva1, promenljiva2, … promenljivan;
```

Primer:

```
union automobil
{
      char[30] proizvodjac;
      char[20] model;
};
union automobil x;
```

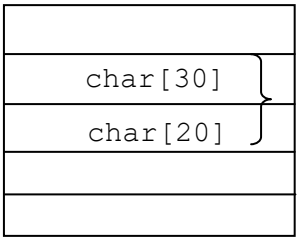
Pristup elementima (članovima) unije vrši se navođenjem imena unije i imena člana razdvojenih tačkom "." ili "->" za pokazivačku promenljivu.

Primer: `x.proizvodjac; x.model;`

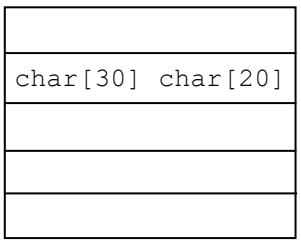
Memorijsko zauzeće

Unija kao i struktura sadži članove različitog tipa, ali kod strukture svaki član ima zasebnu memorijsku lokaciju. Kod unije svi članovi dele istu memorijsku lokaciju. Memorijska lokacija je dovoljno "široka" da u nju stane najširi član unije. Osnovna svrha unije je ušteda memorijskog prostora.

Struktura:



Unija:



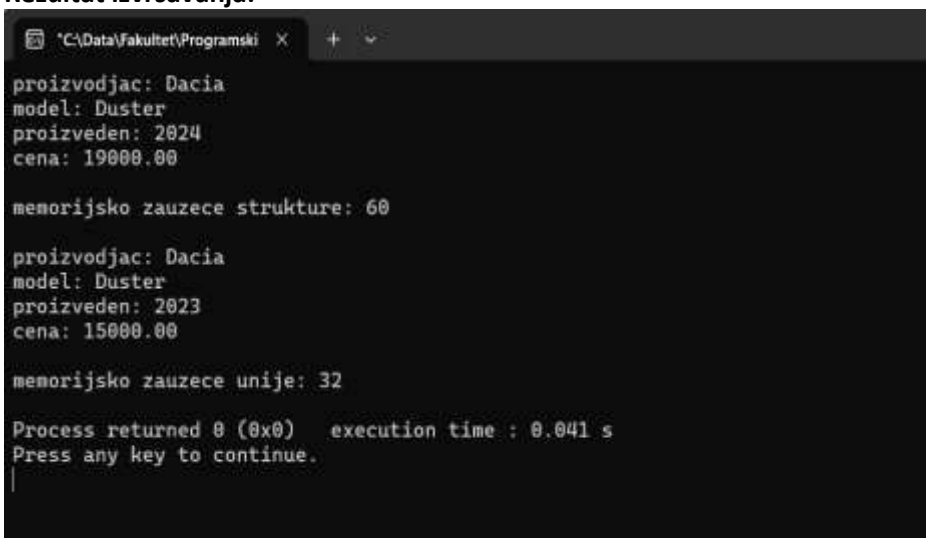
Osobine unije:

- Operacije nad unijom i strukturom su kopiranje i dodela vrednosti;
- Strukture tipa unije se mogu pojaviti kao argumenti funkcije odnosno kao vrednost koju funkcija vraća;
- Ne mogu se porediti;
- Ako treba da se obrađuje veći broj struktura istog tipa, može se koristiti niz unija;
- Moguća je i deklaracija niza sa inicijalizacijom u kojoj nije neophodno navođenje broja elemenata niza tipa unija.

Primer1:

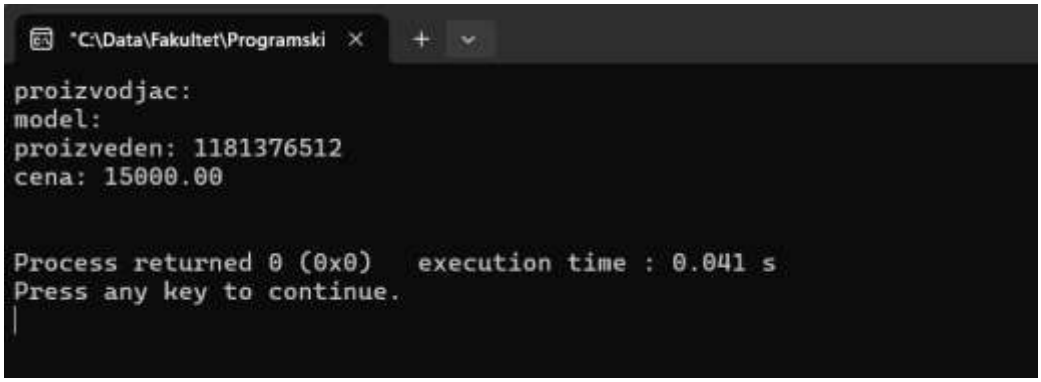
```
#include <stdio.h>
#include <string.h>

struct automobil1
{
char proizvodjac[30];
char model[20];
int godproizvodnje;
float cena;
} auto1;
union automobil2
{
char proizvodjac[30];
char model[20];
int godproizvodnje;
float cena;
}auto2;

int main( )
{
strcpy(auto1.proizvodjac, "Dacia");
strcpy(auto1.model, "Duster");
auto1.godproizvodnje=2024;
auto1.cena=19000;
printf("proizvodjac: %s\n", auto1.proizvodjac);
printf("model: %s\n", auto1.model);
printf("proizveden: %d\n", auto1.godproizvodnje);
printf("cena: %.2f\n\n", auto1.cena);
printf("memorijsko zauzece strukture: %d\n\n", sizeof(auto1));

strcpy(auto2.proizvodjac, "Dacia");
printf("proizvodjac: %s\n", auto2.proizvodjac);
strcpy(auto2.model, "Duster");
printf("model: %s\n", auto2.model);
auto2.godproizvodnje=2023;
printf("proizveden: %d\n", auto2.godproizvodnje);
auto2.cena=15000;
printf("cena: %.2f\n\n", auto2.cena);
printf("memorijsko zauzece unije: %d\n", sizeof(auto2));
return 0;
}
```

Rezultat izvršavanja:



Primer2 - neispravna upotreba elemenata unije**:**

```
#include <stdio.h>
#include <string.h>

union automobil2
{
char proizvodjac[30];
char model[20];
int godproizvodnje;
float cena;
};

int main( )
{
    union automobil2 auto2;
    strcpy(auto2.proizvodjac, "Dacia");
    strcpy(auto2.model, "Duster");
    auto2.godproizvodnje=2023;
    auto2.cena=15000;
    printf("proizvodjac: %s\n", auto2.proizvodjac);
    printf("model: %s\n", auto2.model);
    printf("proizveden: %d\n", auto2.godproizvodnje);
    printf("cena: %.2f\n\n", auto2.cena);
    return 0;
}
```

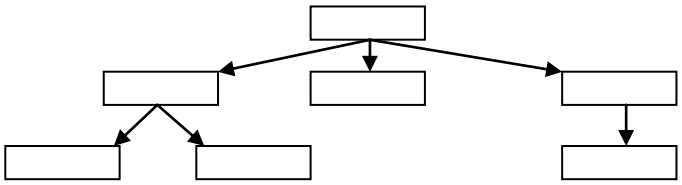


35. DINAMIČKE STRUKTURE PODATAKA

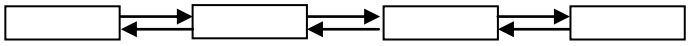
Najčešće korištene dinamičke strukture podataka su: grafovi, liste, stek i red. Osnovne karakteristike dinamičkih struktura podataka jesu to da broj elemenata strukture nije unapred poznat i može se menjati u toku izvršavanja programa.

Grafička predstava dinamičkih struktura podataka:

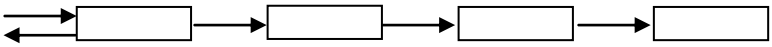
Graf



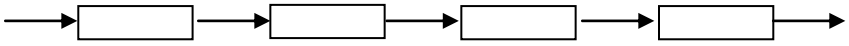
Lista



Stek



Red



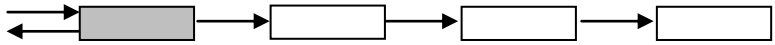
Dinamička struktura podataka tipa graf sadrži jedan koreni element, koji je povezan sa jednim ili više elemenata koji predstavljaju njegove sledbenike. Generisanje novih čvorova u grafu iniciraju različiti operatori (operacije, funkcije, procedure) i koji zavise od semantike problema. Čvorovi se organizuju po nivoima, pri čemu je koreni čvor na nultom nivou. Za svaki čvor se mora znati koji čvor mu je prethodnik i koji čvorovi se mogu dostići iz tog čvora (sledbenici). Koreni čvor nema prethodnika, a postoje i oni čvorovi koji nemaju sledebnike (tzv. lišće stabla, kako se još drugačije graf zove).

Dinamička struktura podataka u obliku jednostruko ili dvostruko spregnute liste sadrži elemente liste, tj. čvorove, koji su povezani pokazivačima na sledeći element (jednostruko spregnuta) i pokazivačima na prethodni element (dvostruko spregnuta). Da bi se moglo pristupati elementima liste potreban je pokazivač na prvi elemenat liste, koji se naziva glava liste. Sa listama se mogu uraditi sledeće operacije:

- inicijalizacija liste,
- unos novog elementa,
- prikaz elemenata iz liste,
- brisanje elementa iz liste,
- brisanje cele liste.

Dinamička struktura podataka tipa stek (engl. "Stack") – elementi strukture imaju osobinu da onaj element koji je poslednji pridružen (ušao u stek) strukturi mora prvi da izađe iz strukture, pa zbog toga sledi i drugi naziv – stog. Ova karakteristika je na engleskom jeziku akronim za naziv: **L**ast **I**n **F**irst **O**ut (LIFO). Mogu se implementirati pomoću nizova i uređenih lista.

Grafički prikaz strukture tipa stek:



Operacije sa stekom su:

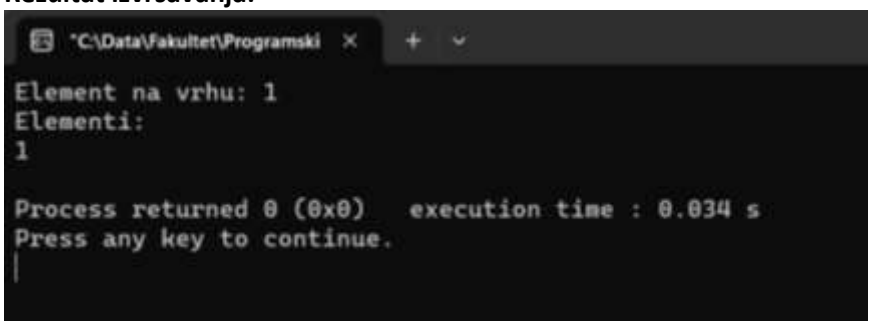
- Push() – dodavanje elementa u stek.
- Pop() – brisanje elementa iz steka.
- Peek() – vraćanje poslednjeg elementa.
- IsEmpty() – provera da li je stek prazan.
- IsFull() – provera da li je stek pun.

Primer programa koji ilustruje strukturu tipa stek i operacije, implentirane kao niz:

```
#include <stdio.h>
int MAXSIZE = 7;
int stack[7];
int top = -1;

/* provera da li je stek prazan */
int isempty()
{
   if(top == -1)
      return 1; //prazan
   else
      return 0;
}

/* provera da li je stek pun */
int isfull()
{
   if(top == MAXSIZE - 1)
      return 1; //pun
   else
      return 0;
}

/* vracanje poslednjeg elementa */
int peek()
{
   return stack[top];
}

/* brisanje iz steka */
int pop()
{
   int data;
   if(!isempty())
    {
      data = stack[top];
      top = top - 1;
      return data;
    }
    else
    {
      printf("Nema podataka, stek je prazan.\n");
    }
}
```

```
/* upis u stek */
int push(int data)
{
   if(!isfull())
    {
      top = top + 1;
      stack[top] = data;
    }
    else
    {
      printf("Ne mozete dodati podatak, stek je pun.\n");
    }
}

/* Glavni program */
int main()
{
   push(1);
   //push(2);
   //push(7);
   //push(10);
   //push(15);
   //push(26);
   //push(38);
   //push(39);
   printf("Element na vrhu: %d\n" ,peek());
   printf("Elementi: \n");

   // print stack data
   while(!isempty())
   {
      int data = pop();
      printf("%d\n",data);
   }
   return 0;
}
```

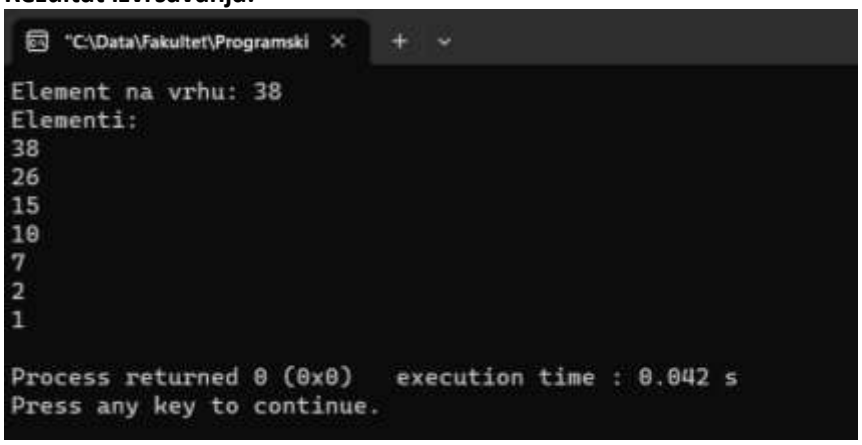
Rezultat izvršavanja:



Nakon dodavanja novih elemenata preko isključivanja komentara za naredbe:

```
push(2);
push(7);
push(10);
push(15);
push(26);
push(38);
```

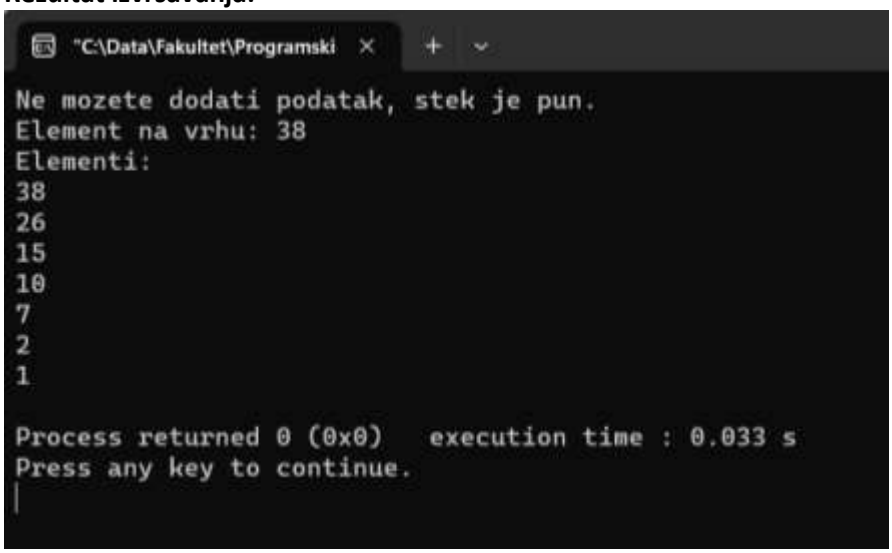
Rezultat izvršavanja:



Nakon dodavanja još jednog elementa, osmog po redu, isključivanjem poslednjeg komentara za naredbe:

```
push(39);
```

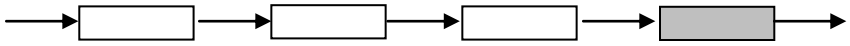
Rezultat izvršavanja:



36. RED

Dinamička struktura podataka tipa red (engl. "Queue") - elementi strukture imaju ključnu osobinu da onaj element koji je prvi pridružen (ušao u red) strukturi, mora prvi i da izađe iz strukture. Ova karakteristika je na engleskom jeziku akronim za naziv: **F**irst **I**n **F**irst **O**ut (FIFO).

Grafički prikaz strukture tipa red:



Operacije za red su:

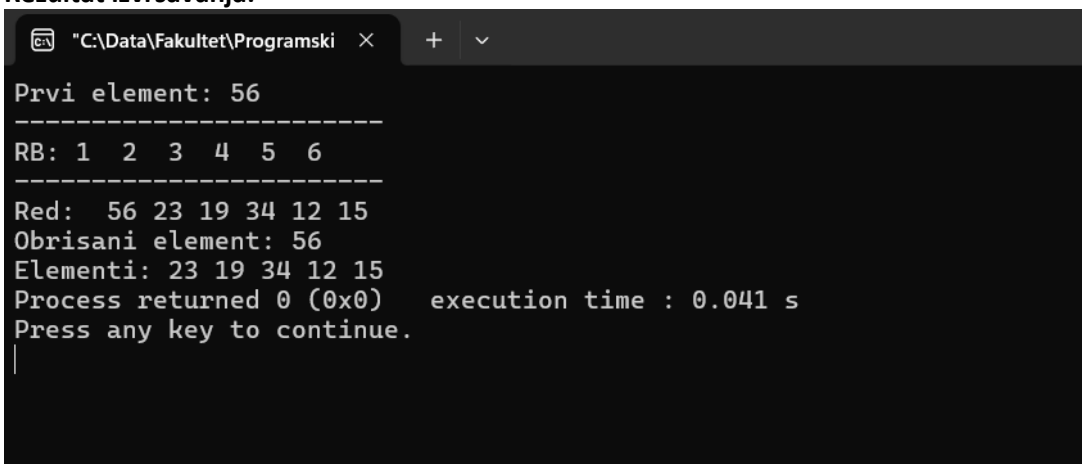
- Insert() – dodavanje elementa u red.
- Remove() – brisanje elementa iz reda.
- Peek() – vraćanje prvog elementa.
- IsEmpty() – provera da li je red prazan.
- IsFull() – provera da li je red pun.
- Size() – veličina, broj elemenata reda.

Primer programa koji ilustruje strukturu tipa red:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

#define MAX 6

int intArray[MAX];
int front = 0;
int rear = -1;
int itemCount = 0;

int peek()
{
   return intArray[front];
}

bool isEmpty()
{
   return itemCount == 0;
}

bool isFull()
{
   return itemCount == MAX;
}

int size()
{
   return itemCount;
}

void insert(int data)
{
   if(!isFull())
   {
      if(rear == MAX-1) {
         rear = -1;
```

```
      }
      intArray[++rear] = data;
      itemCount++;
   }
}

int removeData()
{
   int data = intArray[front++];
   if(front == MAX)
    {
      front = 0;
    }
   itemCount--;
   return data;
}

int main()
{
   // dodavanje elemenata
   insert(56);
   insert(23);
   insert(19);
   insert(34);
   insert(12);
   insert(15);
   insert(11);
   printf("Prvi element: %d\n",peek());
   printf("------------------------\n");
   printf("index: 1  2  3  4  5  6 \n");
   printf("------------------------\n");
   printf("Red:  ");
   while(!isEmpty())
    {
      int n = removeData();
      printf("%d ",n);
    }
  if(isFull())
    {
      printf("Red je pun!\n");
    }
   // brisanje prvog elementa
   int num = removeData();
   printf("\nObrisani element: %d\n",num);
   int i = 1;
   printf("Elementi: ");
   for(i;i<=MAX-1;i++)
    {
      printf("%d ",intArray[i]);
    }
}
```

Rezultat izvršavanja:



KORIŠTENA LITERATURA:

1. A. Hansen: Programiranje na jeziku C – Potpuni vodič za programski jezik C, Mikro knjiga, Beograd, 2000.
2. C.H. Pappas, W.H. Murray: C/C++ vodiš za programere, Mikro knjiga, Beograd, 1996.
3. M. Jurak: Programski jezik C, 2003/2004.
4. D. Ritchie, W. Kernighan: Programski jezik C, CET, Beograd, 2003.
5. C.L. Tondo, S.E. Gimpel: Programski jezik C rešenja zadataka, drugo izdanje, CET, Beograd, 1989.
6. V. Ćirić: Uvod u programiranje I programski jezik C, Univerzitet u Nišu, Elektronski fakultet, 2014.
7. B. Markoski: Praktikum rešenih zadataka iz programskog jezika C, Univerzitet u novom Sadu, Tehnički fakultet "Mihajlo Pupin", Zrenjanin, 2013.
8. B. Markoski: Interni materijal za spremanje ispita iz Programskih jezika, Univerzitet u novom Sadu, Tehnički fakultet "Mihajlo Pupin", Zrenjanin, 2021/2022.
9. N. Jovanović: Uvod u programiranje, Viša poslovna škola, Blace, 2005.
10. D. Malbaški: Algoritmi i strukture podataka, Univerzitet Educons, Sremska Kamenica, 2015.
11. D. Malbaški: Programski jezici i strukture podataka, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad.
12. B. Stojanović: Strukture podataka i algoritmi 2, Prirodno matematički fakultet Kragujevac, 2020.
13. A. Kelley, I. Pohl: A Book on C, Addison-Wesley, 1998.
14. N.M. jovanović: Uvod u programiranje, Viša poslovna škola, Blace, 2005.
15. T. Bailey: An Introduction to the C Programming Language and Software Design, Draft 0.6 (July 12, 2005).
16. https://slidetodoc.com/lekcija-01-uvodna-razmatranja-uvod-u-c-miljan/ M. Milošević, Uvod u C.
17. http://stackoverflow.com, public platform for software development.
18. http://poincare.matf.bg.ac.rs/~gordana//programiranjeI/Programski_Jezik_C_pokazivaci.pdf Programski jezik C – pokazivači, Matematički fakultet, Beograd.
19. http://tutorialspoint.com C Programing tutorial.
20. D. Drakulić: Osnove programskog jezika C sa zbirkom zadataka – skripta.
21. R. Goić: Programski jezik C - bilješke s predavanja, privremeni i nepotpuni materijali, Split, 2002.