Programovací jazyk C

Otázky k ústní maturitě

Obsah

- Datové typy
- Větvení a cykly
- Pole a struktury
- Soubory (definice, použití)
- Funkce (parametry, použití)

Datové typy

Datový objekt je údaj v operační paměti

a je nutné sdělit jakého typu je.

Datový typ označuje druh dat.

Proměnná je paměťový prostor

pro uložení údaje určitého datového typu.

Jméno studenta	textový řetězec	Jmeno

Datové typy

- Při vytváření (definici) proměnné musíme stanovit její datový typ

Datové typy

- Jednoduché
  - Typ žádné hodnoty
  - Celočíselné datové typy
  - Reálné datové typy

Typ žádné hodnoty

- Void

Použití:

  - Podprogram nic nevrací
  - U tzv. ukazatelů

Celočíselné datové typy

signed	**unsigned**		**Počet bajtů**
int	int	celé číslo	4
short int	short int		2
long int	long int		4
char	char	znak	1

Pro 1 bajt rozsah hodnot

-128 až 127 0 až 255

Reálné datové typy

S pohyblivou řádovou čárkou		**Počet bajtů**
float	jednoduchá přesnost	4
double	dvojitá přesnost	8
long double	zvýšená přesnost	10

Příklady definicí

int i, j; int n=5;

signed long pozice;

unsigned char znak;

char zn='x'; float r=2.44;

long double vysledek;

Předefinování f = (int)g / h;

Vstup, výstup a datový typ

Například:

scanf(“%d %f %c”, &cele_cislo, &realne, &znak);

printf("Prumer je %4.2f",prumer);

Větvení

if (log. výraz) příkaz_1; else příkaz_2;

```
if ((x <= hm)&&(x >= dm))
    printf ("lezi v intervalu\n");
 else
     printf ("cislo je mimo interval\n");
```

Nebo skupina příkazů {}

Větvení

switch

```
switch (i){
      case 6:printf("Sobota\n");
            break;
     case 7:printf("Nedele\n");
            break;
     default:printf("Ne volny den\n");
           break;//neni nutny
   }
```

i je celočíselná hodnota

Cykly

Se známým počtem průchodů

for (počáteční_výraz; koncový_výraz; iterace);

```
for ( i=1; i <= 10; i++ ) {
        printf (”%d. \n", i);
        }
```

Cykly

S podmínkou na začátku

while (výraz)
příkaz;

```
while (i < 5)
    i = i+1;
```

na konci

do{
příkazy;
} while(výraz);

```
do{
    i = i+1;    // i++;
} while (i < 5);
```

Cykly

break; ukončí cyklus

continue; skočí na konec cyklu a pokračuje další iterací

Pole

- Více prvků stejného datového typu
  - Statické
  - Dynamické

- Pracovali jsme se statickým
  - Jednorozměrným
  - Vícerozměrným

Pole

int a[5]; // deklarace 5-ti prvkového pole

scanf("%d",&a[i]);

---

char jmeno[50]; //pole znaků je řetězec (string)

scanf("%s",jmeno);

---

int m[4][5]; //dvourozměrné pole

scanf("%d",&m[i] [j]);

1. prvek pole má index **0**

Struktura

- Jeden identifikátor obsahuje více údajů různých typů

```
typedef struct
{
   char typ[20], SPZ[8];
   int vykon;
}Tauto;

...
```

```
Tauto vuz;
...
scanf("%s", vuz.typ);
scanf("%s“,vuz.SPZ);
scanf(„%d“,&vuz.vykon);
...
```

Soubory (definice, použití)

Soubor

- označuje **pojmenovanou** sadu **informací** uloženou **na datovém médiu**.
- posloupnost bajtů ukončena speciální kombinací EOF

Soubory - rozdělení

- Textové - lze je prohlížet v lib. text editoru
  obsahuje řádky textu

- Binární – běžně nečitelný
  obsahuje data tak,
  jak jsou uloženy v paměti počítače

Soubory definice

Proměnná typu soubor

FILE *f; //* znamená ukazatel na soubor

Soubory - otevření

f = **fopen**("soubor1.txt", "r");

r – read **w** – write **a** - append

```
if ((f = fopen("soubor1.txt", "r")) == NULL) {
    printf("Soubor se nepodarilo otevrit\n");
    system("pause");
    return 1;
  }
```

Soubory - uzavření

fclose(f);

```
if (fclose(f) == EOF) {
    printf("Soubor se nepodarilo uzavrit\n");
    return 1;
  }
```

Soubory – zápis

Textové

fprintf(f,"%d\n",cislo);

putc(znak,f);

Binární

fwrite(&hodnota,sizeof(hodnota),1,f);

počet údajů pro jednu položku dané velikosti

Soubory – čtení

Textové

fscanf(f, "%f", &x);

znak=**getc**(f);

Binární

fread(&hodnota,sizeof(hodnota),1,f);

počet údajů pro jednu položku dané velikosti

Funkce (parametry, použití)

- Funkce je podprogram, který vrací nějakou hodnotu
- Jazyk C nepodporuje procedury, které nevrací hodnotu, ale používá funkce s návratem žádné hodnoty typu void

Knihovní funkce už existují, jsou v knihovnách

např. printf()

Vlastní funkce vytváříme sami

Funkce - parametry

- Pomocí parametrů předáváme např. vstupní informace pro výpočet
- Funkce nemusí mít žádný parametr

  např. main()
- Parametry dělíme na
  - formální
  - skutečné

Funkce - použití

```
#include<stdio.h>
#include<stdlib.h>
int Mensi(int,int);        /* deklarace funkce */
int main()
{
   int a,b;                /* skutečné parametry*/

   printf("Zadej dve cisla: ");
   scanf("%d%d",&a, &b);

   printf("Mensi ze dvou cisel %d a %d je %d\n",a,b,Mensi(a,b));
   system("pause");
   return 0;
}
```

Funkce - použití

```
int Mensi(int x1, int x2)          /* formální parametry*/
{
   return (x1 < x2) ? x1 : x2;
}
```

Funkce může obsahovat lokální proměnné

Rekurzivní funkce

- Volají sami sebe
- Je nutné dobře stanovit ukončovací podmínku – jinak hrozí přeplnění paměti a zatuhnutí programu.

Rekurzivní funkce

```
unsigned long faktorial (int n)
{
   if (n==0)
        return 1;
   else
        return(n*faktorial(n-1));
}
```