10. Процедурно програмиране.

I. Скаларни типове от данни. Логически тип. Числени типове цял и реален.

1. Скаларни типове данни:
   - реален
   - цял
   - изброен
   - псевдоним
   - символен
   - указател
   - булев

2. Логически тип (булев тип).

```
bool a;
bool b = false;
```

а) логически оператори
   - логическо умножение - конюнкция - "and" ; "&&"
   - логическо събиране - дизюнкция - "or" ; "||"
   - логическо отрицание - "not" ; "!"

б) оператори за сравнение: "==", ">=", "<=", "!=", "<", ">"

в) въвеждане: не се въвежда с оператора ">>".

г) извеждане: cout << <булев-израз>;

д) булеви изрази: това са правилата за получаване на булева стойност. ① прилагането на булевите оператори над бул. изр. е бул. изр.
② прилагането на опер. за срв. над бул. изр. е бул. изр.

3. Числени типове

а) целочислени типове: int е целочислен тип.

1) аритметични оператори
   - унарни - записват се преди или след единствения аргумент
   - бинарни - два аргумента: + - * / %
   - логически оператори - конюнкция, дизюнкция, отрицание
   числа ≠ 0 true; 0 - false

- оператори за сравнение "==", "!=", "<=", ">=", "<", ">"

2) Вградени функции: в C++ обръщението към вградени функции има следния синтаксис:

& <име_на_функция>(<израз>, <израз>, ..., <израз>)

* връща стойност от типа на функцията.

пр. abs(x) връща |x|.

abs(-10) = 10.

3) други целочислени типове - short int
- unsigned short int
- long int
- unsigned long int
- unsigned int

б) реални типове - пр. double

1) аритметични оператори
- унарни оператори "+"; "-" префиксни, с 1 аргумент.
- бинарни оператори - инфиксни, с 2 аргумента: +; -; *; /
- логически оператори - конюнкция, дизюнкция, отрицание
  числа ≠ 0.0 - true ; 0.0 - false.
- оператори за сравнение (==, <, >, >=, <=, !=)

* допълнение: сравнението за равно на две числа: $|x-y|<\varepsilon$, $\varepsilon = 10^{-14}$ double

2) вградени функции (връщат резултат double)

sin(x), cos(x), tan(x), log(x), sqrt(x) ... математически.

3) въвеждане и извеждане cin; cout

4) други реални типове - float (различава се от double по точност от ст-ти и заемана памет)

II. Съставни типове от данни. Структура от данни масив. Тип масив

1. Съставни типове от данни
- масив
- вектор
- запис

2. Структура от данни "масив"

а) логическо описание

* масив - масива е крайна редица от фиксиран брой

елементи от един и същи тип. Всеки елемент е свързан с долен срезиндекс. Масива е статична структура от данни.

б) физическо представяне:

- за всеки елемент се заделя част от паметта.

в) тип масив.

- В С++ структурата от данни масив се разглежда като крайна редица от елементи от един и същ тип с пряк достъп до всеки елемент, осъществяващ се срезиндекс оценяващ от-ти започващи от 0 и нарастващи с 1 до указана горна граница. Дефинира се от програмиста.

г) дефиниране на масив:

int [5] - дефинира масив от 5 елемента с индекс 0-4 тип int.

ф) множество от стойности

int m[4]={1, 2, 3, 4, 5}

int m[5] = {1, 2, 3} е еквивалентно на int m[5]={1, 2, 3, 0, 0}

е) многомерни масиви. - масив, базов елемент на който е едномерен масив с фиксирана двумерен.

int [3] [5]

III Тип указател - дефиниране, основни операции. Указателна аритметика.

1. Тип указател.

int i = 1024;

- променлива с име i;
- променливата i е от тип int;
- стойността ѝ (value) е 1024 и именува място от паметта (&value) с размер 4 байта, като value е адреса на първия байт на това място

Синтаксис на ~~тип указател~~ адрес: & <име_на_променлива>

<име_на_променлива> е всяко дефинирано преди променлива

Адрес

Указател на променлива

Семантика: &<име-на-променлива> намира адреса на <име-на-променлива>

Адреса се извежда с: cout<<&i;

Указателен тип T : T*

int : int*

{a,b,c} : {a,b,c}*

Променлива от тип указател int* a;

Извличане на съдържанието на указател: с оператор *

* <променлива_от_тип_указател>

->извлича стойността на адреса, записан в <променлива_от_тип_указ>

```
int i = 12;
int* p = &i   // p е инициализирано с адреса на i
double *q = NULL  // q е инициализирано нулев вид указател
double x = 1.56;
double *r = &x; // r е инициализирано адреса на x
```

*p е 12

*r е 1.56

(адреса)

2. Указателна аритметика.

+; -; ++; --; ==; !=; >; >=; <; <=

а) въвеждане cin << - не е възможно

б) извеждане cout >> възможно

IV. Указатели и масиви

1. **Указатели и едномерни масиви.**

Друг запис на a[i] (i=0,1,...,n-1) е *(a+i)

Именуваните масиви са константни указатели

2. Указатели и двумерни масиви

- името на двумерен масив е указател към първия елемент на едномерен масив от константни указатели.

```
int a[10][20]
a[i] == *(a+i)
a[i][j] == (*(a+i))[j] == *(*(a+i)+j)
```

3 Указатели и низове

Низовете са масиви от символи

Името на променлива от тип низ е константен указател.

```
char* pstr = str;    // pstr е указател към низа str.
char str[] = "vid"   // str е константен указател.
```

V Функции. Дефиниране на ф-я. Обръщение към ф-я.

1. Дефиниране на функция: заглавие и тяло.

[<модификатор>] [<тип-на-функцията>] <име-на-функцията> ↵
(<формални параметри>) {<тяло>}

```
#include
   static void main (string)
void printtext(void)
  { cout<<" C++ Progr lang \n"
    cout<<" B. Steoustrup \n";
    return;
  }
```

2. Видове функции.

а) функции, които връщат стойност

б) функции, които не връщат стойност - void (при дефиниция указва, вместо тип, се пише void).

3. Оператор Return. return [<израз>] оп ц.

запазена дума → return

правилен изразтип <тип-на-функцията> или съвместимо нему; при тип void, <израз> се пропуска. → [<израз>]

4. Обръщение към функция

<обръщение_към_функция> ::=
<име_на_функция>() |
<име_на_функция>(void) |
<име_на_функция>(<фактически_параметри>)

→ толкова на брой, колкото и формалните.

а) Свързване на формалните с фактическите параметри.
- първи формален се свързва с първи фактически,
- втори формален се свързва с втори фактически и т.н.

Свързването се реализира по различни начини в зависимост от вида на формалния параметър.

1) формален параметър - стойност - в този случай се намира стойността на съответния фактически параметър. (толкова памет, колкото тя изисква.)

2) формален параметър - указател. В този случай в стековата рамка на функцията на фактическия параметър се отделят 4В, в които се записва стойността на фактическия параметър, който е адрес на променлива.

3) формален параметър-псевдоним. - формалния параметър-псевдоним се свързва с адреса на фактическия, не се отделя памет.

VII. Функции. Масив като формален параметър.

- Предава се по адрес (адреса на първия елемент и се работи с оригиналните елементи на масива) => Промени на елементите на масива в тялото на функцията ще се отразят и в извикващата функция.

- Едномерен масив като формален параметър на функция се декларира по един от трите начина:
  - като указател – void funct (int *x)
  - масив с определен размер - void funct (int x[5])
  - масив без определен размер - void funct (int x[ ])